Volume 29 Number 2 June 2005 ISSN 0350-5596

Informatica

An International Journal of Computing and Informatics

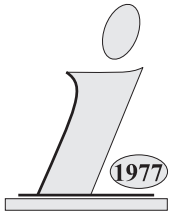
Special Issue:

Ant Colonies and Multi-Agent Systems

Guest Editors:

Nadia Nedjah

Luiza de Macedo Mourelle



The Slovene Society Informatika, Ljubljana, Slovenia

EDITORIAL BOARDS, PUBLISHING COUNCIL

Informatica is a journal primarily covering the European computer science and informatics community; scientific and educational as well as technical, commercial and industrial. Its basic aim is to enhance communications between different European structures on the basis of equal rights and international refereeing. It publishes scientific papers accepted by at least two referees outside the author's country. In addition, it contains information about conferences, opinions, critical examinations of existing publications and news. Finally, major practical achievements and innovations in the computer and information industry are presented through commercial publications as well as through independent evaluations.

Editing and refereeing are distributed. Each editor from the Editorial Board can conduct the refereeing process by appointing two new referees or referees from the Board of Referees or Editorial Board. Referees should not be from the author's country. If new referees are appointed, their names will appear in the list of referees. Each paper bears the name of the editor who appointed the referees. Each editor can propose new members for the Editorial Board or referees. Editors and referees inactive for a longer period can be automatically replaced. Changes in the Editorial Board are confirmed by the Executive Editors.

The coordination necessary is made through the Executive Editors who examine the reviews, sort the accepted articles and maintain appropriate international distribution. The Executive Board is appointed by the Society Informatika. Informatica is partially supported by the Slovenian Ministry of Science and Technology.

Each author is guaranteed to receive the reviews of his article. When accepted, publication in Informatica is guaranteed in less than one year after the Executive Editors receive the corrected version of the article.

Introduction

Instead of designing complex and centralized systems, researchers rather prefer to work with many small and autonomous agents. The agents mimic the ant's behavior within an ant colony. Each one acting on the simplest of rules, these many agents can solve very complex problems known as hard problems. Generally, such multi-agent systems are used as search and optimization tools.

This special issue of the *Informatica - International Journal of Computing and Informatics* is focused on ant colonies and multi-agent systems. It includes seven contributions that describe new methods and experiences for multi-agent implementations of aspects of artificial life, ant colony and swarm intelligence.

The first paper is entitled "Investigating Strategic Inertia Using OrgSwarm" and was proposed by *Anthony Brabazon, Arlindo Silva, Tiago Ferra de Sousa, Michael O'Neill, Robin Matthews and Ernesto Costa*. The study describes a novel simulation model, called *OrgSwarm*, of the process of strategic adaptation. In this paper, strategic adaptation is conceptualized as a process of adaptation or search, on a landscape of strategic possibilities, by a population of profit-seeking organizations.

The second paper is entitled "Towards Improving Clustering Ants: An Adaptive Ant Clustering Algorithm" and was proposed by *André L. Vizine, Leandro N. de Castro, Eduardo R. Hruschka, Ricardo R. Gudwin*. The paper introduces and discusses both a progressive vision scheme and pheromone heuristics for the standard ant-clustering algorithm, together with a cooling schedule that improves its convergence properties. The proposed algorithm is evaluated in a number of well-known benchmark data sets, as well as in a real-world bioinformatics dataset.

The third paper is entitled "Efficient Pre-Processing for Large Window-Based Modular Exponentiation Using Ant Colony" and was proposed by *Nadia Nedjah and Luiza de Macedo Mourelle*. The paper exploits the ant colony strategy to finding an optimal addition sequence that allows one to perform the pre-computations in window-based methods with a minimal number of modular multiplications and hence, improves the efficiency of modular exponentiation.

The forth paper is entitled "Max Min Ant System and Capacitated *p*-Medians: Extensions and Improved Solutions" and was proposed by *Fabrício Olivetti de França, Fernando J. Von Zuben, Leandro Nunes de Castro*. The work introduces a modified MAX MIN Ant System (MMAS) designed to solve the Capacitated *p*-Medians Problem (CPMP). It presents the most relevant steps towards the implementation of an MMAS to solve the CPMP, including some improvements on the original MMAS algorithm, such as the use of a density model in the information heuristics and a local search adapted from the un-capacitated *p*-medians problem.

The fifth paper is entitled "Application of Ant-based Template Matching for Web Documents Categorization" and was proposed by *Siok Lan Ong, Weng Kin Lai, Tracy S. Y. Tai, Choo Hau Ooi and Kok Meng Hoe*. The paper examines the direct implementation of a template based on a Gaussian Probability Surface to supervise these homogeneous multi-agents to form clusters within a specified dropping zone.

The sixth paper is entitled "Efficient and Scalable Communication in Autonomous Networking using Bio-inspired Mechanisms – An Overview" and was proposed by *Falko Dressler*. In this paper, the author demonstrates the possibilities which evolve by the application of cell biology for computer networking. With the focus on autonomous networking, the combination with methodologies known from swarm intelligence is evaluated. The author shows the capabilities of this combination and derive destinations and goals for self-organization in communication networks showing a more efficient and scalable behavior.

The seventh paper is entitled "Model Checking Multi-Agent Systems" and was proposed by *Mustapha Bourahla and Mohamed Benmohamed*. In this paper, the authors show how a well known and effective verification technique, model checking, can be generalized to deal with multi-agent systems. The paper explores a particular type of multi-agent system, in which each agent is viewed as having the three mental attitudes of belief, desire and intention.

Nadia Nedjah and Luiza de Macedo Mourelle

Investigating Strategic Inertia Using OrgSwarm

Anthony Brabazon, Arlindo Silva, Tiago Ferra de Sousa, Michael O'Neill, Robin Matthews, Ernesto Costa
University College Dublin, Ireland.
anthony.brabazon@ucd.ie

Escola Superior de Tecnologia, Instituto Politecnico de Castelo Branco,Portugal.
arlindo@est.ipcb.pt

University of Limerick, Ireland.
michael.oneill@ul.ie

Kingston University, London.
r.matthews@kingston.ac.uk

Centro de Informatica e Sistemas da Universidade de Coimbra, Portugal.
ernesto@dei.uc.pt

Keywords: Particle swarm, Organizational adaptation, OrgSwarm

Received: July 26, 2004

*This study describes a novel simulation model (**OrgSwarm**) of the process of strategic adaptation. Strategic adaptation is conceptualized as a process of adaptation (search), on a landscape of strategic possibilities, by a population of profit-seeking organizations. Unfortunately, the characteristics that make organizations coherent and viable such as organizational structure and shared organizational culture, also create strategic inertia, potentially limiting the ability of organizations to adapt. This study examines the impact of strategic inertia on the adaptive potential of organizations. The simulation results suggest that a degree of strategic inertia can assist rather than hamper adaptive efforts in static and slowly changing strategic environments.*

Povzetek: Predstavljen je OrgSwarm, nov model procesa strateškega prilagajanja.

1 Introduction

There are parallels between biological and social systems. In both, individuals within a larger population are attempting to appropriate scarce resources, or *to earn a living*, in a dynamic environment. Entities in these systems typically alter their 'strategies' over time in an attempt to improve their success. In an organizational setting, a strategy consists of a choice of what activities the organization will perform, and choices as to how these activities will be performed [36]. These choices define the strategic configuration of the organization. Recent work by [28] and [38] has recognized that strategic configurations consist of interlinked individual elements (decisions), and have applied general models of interconnected systems such as Kauffman's NK model to examine the implications of this for processes of organizational adaptation.

Following a long-established metaphor of adaptation as search [46], strategic adaptation is considered in this study as an attempt to uncover peaks on a high-dimensional strategic landscape. Some strategic configurations produce high profits, others produce poor results. The search for good strategic configurations is difficult due to the vast (combinatorial) number of configurations, uncertainty as to the nature of topology of the strategic landscape faced by an organization, and changes in the topology of this landscape over time. Despite these uncertainties, the search process for good strategies is not blind. Decision-makers receive feedback on the success of their current and historic strategies, and can assess the payoffs received by the strategies of their competitors [26]. Hence, certain areas of the strategic landscape are illuminated.

Organizations do not exist in isolation, but interact with, and receive feedback from, their environment. Their efforts at strategic adaption are guided by social as well as individual learning. Good ideas discovered by one organization disseminate over time. One model combining both individual and social learning which has attracted significant interest in recent years is that of *Particle Swarm Optimization* (PSO) [21], [25]. Particle swarm research has been concentrated in two broad areas, the application and study of PSO as an optimizing tool, and the application of PSO as a model of social and cultural adaptation. This paper adopts the second of these perspectives, and adapts the canonical PSO to create a plausible model of the process of strategic adaptation.

Although the particle swarm model has been applied to a variety of problems in the fields of engineering [1], chemistry [34], medicine and psychology [25], as yet it has not been applied to the domain of organizational science. This

paper introduces the model to this domain, and utilizes it to examine the impact of differing degrees of strategic inertia on the adaptive capabilities of a population of organizations.

1.1 Structure of paper

This contribution is organized as follows. Section 2 provides a short discussion of prior literature in the domain of strategic adaptation in order to provide a number of perspectives on strategic inertia. Section 3 incorporates an introduction to the canonical Particle Swarm algorithm (PSA),[1] followed by a description of the simulation model in Section 4. Section 5 outlines the results of the simulations and finally, conclusions and future work are discussed in Section 6.

2 Strategic Adaptation

Strategic adaptation and strategic inertia are closely linked. If strategic adaptation is problematic, inertia is a possible cause. A substantial literature has emerged on strategic adaptation. This, along with its implications for strategic inertia, is outlined below.

Two polar views exist concerning the ability of organizations to adapt their strategic configuration. Adaptationists or advocates of strategic choice [35], [40], [31], broadly consider that managers or dominant coalitions in organizations scan the environment for current and future opportunities and threats, formulate strategic responses and adjust organizational activities and structure appropriately [10]. Therefore, strategic direction and organizational form are determined by managers, and market selection processes act to maintain organizations which are good adaptors. Under this perspective, an organization's fate is largely in its own hands, and hence strategic inertia is considered to represent a challenge rather than a roadblock to strategic adaptation efforts. The adaptationist argument presupposes that organizations are capable of adapting at least as fast as their environment changes [31], [30]. If firms are incapable of responding to environmental changes in a similar timescale, adaptation (or learning) processes will not enhance organizational survival [13]. The current practitioner interest in 'change management' [16] exemplifies the belief that even substantial strategic adaptation is possible.

In contrast, the population ecology school [12] proposes an alternative view on organizational-environment relations. This school of thought allows that organizations have some ability to adapt to environmental change and notes that '*leaders of organizations do formulate strategies and organizations do adapt to environmental contingencies*' [12] (p. 930). However, it is argued that the ability of firms to accurately and consistently adapt in a world of high uncertainty, where connections between means and ends are unclear is doubtful [13], [9]. Although selection processes select the most fit organizations in a given environment for continued survival, population ecologists contend that an organization's fitness primarily arises because of good initial strategic choices, or luck, rather than reflecting post-founding adaptation [2]. Advocates of the population ecology school suggest that the ability of organizations to adapt is highly constrained because of their inherent inertia. This inertia stems from two sources, *imprinting forces*, and as a *consequence of market selection forces*.

[1] The term PSA is used in place of PSO (Particle Swarm Optimization) in the remainder of this paper, as the object of the paper is not to develop a tool for optimizing, but to apply the swarm metaphor as a model of organizational adaptation.

2.1 Imprinting Forces

Imprinting forces [4] combine to define and solidify the strategic configuration of a newly formed organization. These forces include the dominant initial strategy pursued by the organization, the skills / prior experience of the management team, and the distribution of decision-making influence in the organization at time of founding [4]. These forces influence the initial choice of organizational strategy. As consensus concerning the strategy emerges, it is imprinted on the organization through resource allocation decisions [42]. The imprinting leads to inertia by creating sunk costs, internal political constraints, and a rigid organizational structure. Over time this inertia intensifies due to the formation of an organizational history which creates barriers to industry exit, and legitimacy issues if adaptation is suggested [12]. The resulting inertia serves to circumscribe the organization's ability to adapt its strategy in the future. The initial imprinting determines the basin of attraction in which the organization is located on the strategic landscape. Imprinting also occurs as relationships are built up with suppliers and customers [43]. The creation of a web of these relationships can serve to constrain the range of strategic alternatives in the future, as strategic moves which dramatically disrupt the web are less likely to be considered.

2.2 Market-Selection Forces

The discussion of strategic inertia was extended by [13] who posited that inertia is also created as a natural *consequence* of the market-selection process, claiming that '*selection processes tend to favor organizations whose structures are difficult to change.*' (p. 149). The basis of this claim is that organizations which can produce a good or service reliably (consistently of a minimum quality standard) are favored for trading purposes by other organizations, and therefore by market selection processes. The routines required to produce a product or service reliably, tend to lead to structural inertia, as the construction of routines to achieve this leads to an increase in the complexity of the patterns of links between organizational subunits [13] & [27]. Building on this point, it can be posited that more efficient organizations are likely to exhibit inertia. As organizations seek better environment-structure congruence,

their systems become increasingly specialized and interlinked, making changes to their activities become costly and difficult. Structural inertia is rooted in the size, complexity and interdependence of the firm's structures, systems, procedures and processes [45]. Theoretical support for these assertions, that increasing organizational complexity can make adaptation difficult, is found in [19] and [38], as the heightened degree of interconnections between activities within the firm will increase the 'ruggedness' of the strategic landscape faced by an organization.

The arguments that organizations are subject to strategic inertia also finds resonance in the literature concerning organizational learning and organizational memory. The preference of organizations to continue to pursue activities similar to those undertaken in the past has been widely noted [14], [32], as has the cumulative nature of organizational learning [33].

In summary, strategic inertia can arise from a variety of sources, and the general consensus in organizational literature is that its existence poses clear difficulties for strategic adaptation by organizations.

3 Particle Swarm Algorithm

This section provides an introduction to the Particle Swarm algorithm (PSA). A full description of this algorithm and the cultural model which inspired it is provided in [25]. A Swarm can be defined as '*... a population of interacting elements that is able to optimize some global objective through collaborative search of a space.*' [25](p. xxvii). The nature of the interacting elements (particles) depends on the problem domain, in this study they represent organizations. These particles move (fly) in an n-dimensional search space, in an attempt to uncover ever-better solutions to the problem of interest.

Each of the particles has two associated properties, a current position and a velocity. Each particle also has a memory of the best location in the search space that it has found so far (*pbest*), and knows the location of the best location found to date by all the particles in the population (*gbest*). At each step of the algorithm, particles are displaced from their current position by applying a velocity vector to them. The size and direction of this velocity is influenced by the velocity in the previous iteration of the algorithm (simulates momentum), and the current location of a particle relative to its pbest and gbest. Therefore, at each step, the size and direction of each particle's move is a function of its own history (experience), and the social influence of its peer group. A number of variants of the PSA exist. The following paragraphs provide a description of the basic *continuous* version described by [25]. The algorithm is initially described narratively. This is followed by a description of the particle position-update equations.

3.1 The Algorithm

i. Initialize each particle in the population by randomly selecting values for its location and velocity vectors

ii. Calculate the fitness value of each particle. If the current fitness value for a particle is greater than the best fitness value found for the particle so far, then revise *pbest*

iii. Determine the location of the particle with the highest fitness and revise *gbest* if necessary

iv. For each particle, calculate its velocity according to equation (1)

v. Update the location of each particle

vi. Repeat steps ii - v until stopping criteria are met

Each particle i has an associated current position in the search space x_i, a current velocity v_i, and a personal best position in the search space y_i. During each iteration of the algorithm, the location and velocity of each particle is updated using equations (1) - (5).

To provide intuition on the workings of the algorithm, see figure 1. Each particle i has an associated current position in search space $x(t) = (x_{i1}(t), \ldots, x_{in}(t))$ at time t, a current velocity of $v(t) = (v_{i1}(t), \ldots, v_{in}(t))$, and a pbest position of $y_i(t) = (y_{i1}(t), \ldots, y_{in}(t))$. The position of the particle at time $t+1$ is a determined by $x(t)+v(t+1)$, and $v(t+1)$ is obtained by a stochastic blending of $v(t)$, an acceleration towards gbest (v_{gbest}) and an acceleration towards pbest (v_{pbest}).

Assuming a function f is to be maximized, that the swarm consists of m particles, and that r_1, r_2 are drawn from a uniform distribution in the range (0,1), the velocity update for particle i is as follows:

$$v_i(t+1)=Wv_i(t)+c_1r_1(y_i-x_i(t))+c_2r_2(\hat{y}-x_i(t)) \qquad (1)$$

where $\hat{y}$ is the location of the global-best solution found by all the particles.[2] In every iteration of the algorithm, each particle's velocity is stochastically accelerated towards its previous best position and towards a neighborhood (global) best position. The weight-coefficients c_1 and c_2 control the relative impact of pbest and gbest locations on the velocity of a particle. The parameters r_1 and r_2 ensure that the algorithm is stochastic. A practical effect of the random coefficients r_1 and r_2, is that neither the individual nor the social learning terms are always dominant. Sometimes one or the other will dominate [25]. Although the velocity update has a stochastic component, the search process is not random. It is guided by the memory of past 'good' solutions corresponding to a psychological tendency for individuals to repeat strategies which have worked for them in the past

[2] A variant on the basic algorithm is to use a local rather than a global version of gbest. In the local version, gbest is set independently for each particle, based on the best point found thus far within a *neighborhood* of that particle's current location.

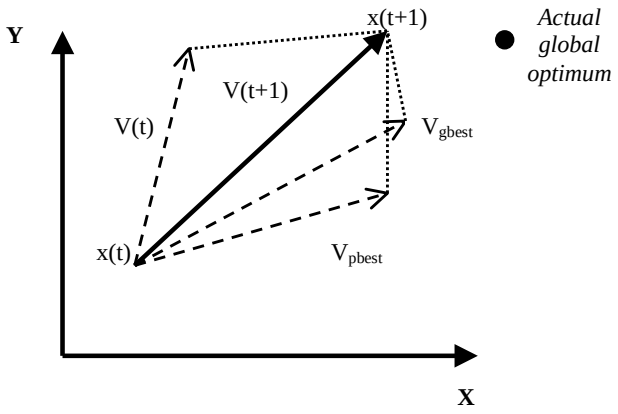


Figure 1: A representation of the particle position-update process.

[22], and by the global best solution found by all particles thus far. W represents a momentum coefficient which controls the impact of a particle's prior-period velocity on its current velocity. Each component of a velocity vector v_i is restricted to a range $[-v_{max}, v_{max}]$ to ensure that individual particles do not leave the search space. The implementation of a v_{max} parameter can also be interpreted as simulating the incremental nature of most learning processes [22]. The value of v_{max} is usually chosen to be $k * x_{max}$, where $0 < k < 1$. Once the velocity update for particle i is determined, its position is updated and pbest is updated if necessary.

$$x_i(t+1)=x_i(t)+v_i(t+1) \tag{2}$$

$$y_i(t+1)=y_i(t) \text{ if, } f(x_i(t))\leq f(y_i(t)), \tag{3}$$

$$y_i(t+1)=x_i(t) \text{ if, } f(x_i(t))>f(y_i(t)) \tag{4}$$

After all m particles have been updated, a check is made to determine whether gbest needs to be updated.

$$\hat{y}\in(y_0,y_1,...,y_m)|f(\hat{y})= \max\ (f(y_0),f(y_1),...,f(y_m)) \tag{5}$$

3.1.1 PSA vs the Genetic Algorithm

It is noted that the PSA bears similarity to other biologically-inspired optimizing algorithms. Like the Genetic Algorithm (GA), it is a population-based algorithm, is typically initialized with a population (swarm) of random solutions, and search proceeds by updating these solution each generation (iteration). Unlike the GA, the move (update) operators are not direct analogs of the genetic operators of mutation and crossover,[3] there is no explicit selection process, and potential solutions are referred to as particles rather than as chromosomes.

The communication (information-sharing) mechanism of the PSA also differs from that of the GA. In the GA, the communication is between two solutions, in the PSA, the communication is between the current solution, its pbest and the gbest. Hence, candidate solutions can 'see' the global best solution found by all particles thus far. The movement of each particle through the search space is influenced by their own previous experience (history) and a wish to move towards the global, best position found thus far by other particles [39].

3.2 The PSA and Social Learning

Despite its simplicity, the PSA is capable of capturing a surprising level of complexity, as individual particles are capable of both individual and social learning. In social settings, individuals are not '*...isolated information-processing entities ...*' [25] (p. xv), but also learn from the experiences of their peers. Social behavior helps individuals to adapt to their environment, as it ensures that they obtain access to more information than that captured by their own senses. Learning in social species is therefore distributed and parallel.

Communication (interactions) between agents (individuals) in a social system may be direct or indirect. An example of the former could arise when two organizations trade with one another. Examples of the latter include:

i. the observation of the success (or otherwise) of a strategy being pursued by another organization, and

ii. *stigmergy* which arises when an organization modifies the environment, which in turn causes an alteration of the actions of another organization at a later time.

The first of these indirect learning mechanisms is included in the canonical PSA, the second can be included through an adaption of the basic model.

[3] It can be argued that the velocity vector update does bear similarity to a recombination operator, being impacted by the location of pbest and gbest [21].

The mechanisms of the basic Particle Swarm model bear *prima facie* similarities to those of the domain of interest, organizational adaptation. It embeds concepts of a population of entities which are capable of individual and social learning. However, the model requires modification before it can employed as a plausible model of organizational adaptation. These modifications, along with a definition of the strategic landscape used in this study are discussed in the next section.

4 OrgSwarm Model

This section describes the simulation model (OrgSwarm) employed in this study [7], [8]. The model can be classed as a multi-agent system (MAS). MASs focus attention on collective intelligence, and the emergence of behaviors through the interactions between the agents. MAS usually contain a world (environment), agents, relations between the entities, a set of activities that the agents can perform, and changes to the environment as a result of these activities [44]. The key components of the simulation model, the landscape generator (environment), and the adaption of the basic Particle Swarm algorithm to incorporate the activities and interactions of the agents (organizations) are described next.

4.1 Strategic Landscape

In this study, the strategic landscape is defined using the NK model [18], [19]. Application of the NK model to define a strategic landscape is not atypical and has support from existing literature in organizational science [28],[38], [15], and related work on technological innovation [29], [20], [41], [37]. The NK model considers the behavior of systems which are comprised of a configuration (string) of N individual elements. Each of these elements are in turn interconnected to K other of the N elements (K<N). In a general description of such systems, each of the N elements can assume a finite number of states. If the number of states for each element is constant (S), the space of all possible configurations has N dimensions, and contains a total of $\prod_{i=1}^{N} S_i$ possible configurations.

In Kauffman's operationalization of this general framework [19], the number of states for each element is restricted to two (0 or 1). Therefore the configuration of N elements can be represented as a binary string. The parameter K, determines the degree of fitness interconnectedness of each of the N elements and can vary in value from 0 to N-1. In one limiting case where K=0, the contribution of each of the N elements to the overall fitness value (or worth) of the configuration are independent of each other. As K increases, this mapping becomes more complex, until at the upper limit when K=N-1, the fitness contribution of any of the N elements depends both on its own state, and the simultaneous states of all the other N-1 elements, describing a fully-connected graph.

If we let s_i represent the state of an individual element i, the contribution of this element (f_i) to the overall fitness (F) of the entire configuration is given by $f_i(s_i)$ when K=0. When K>0, the contribution of an individual element to overall fitness, depends both on its state, and the states of K other elements to which it is linked ($f_i(s_i : s_{i1}, ..., s_{ik})$). A random fitness function (U(0,1)) is adopted, and the overall fitness of each configuration is calculated as the average of the fitness values of each of its individual elements. Therefore, if the fitness values of the individual elements are $f_1, ..., f_N$, overall fitness (F) is:

$$F = \frac{\sum_{i=1}^{N} f_i}{N} \tag{6}$$

Altering the value of K effects the ruggedness of the described landscape, and consequently impacts on the difficulty of search on this landscape [18], [19]. The strength of the NK model in the context of this study is that by tuning the value of K it can be used to generate strategic landscapes (graphs) of differing degrees of local-fitness correlation (ruggedness). The strategy of an organization is characterized as consisting of N attributes [28]. Each of these attributes represents a strategic decision or policy choice, that an organization faces. Hence, a specific strategic configuration s, is represented as a vector $s_1, ..., s_N$ where each attribute can assume a value of 0 or 1 [38]. The vector of attributes represents an entire organizational form, hence it embeds a choice of markets, products, method of competing in a chosen market, and method of internally structuring the organization [38]. Good consistent sets of strategic decisions (configurations), correspond to peaks on the strategic landscape.

The definition of an organization as a vector of strategic attributes finds resonance in the work of Porter [35], [36], where organizations are conceptualized as a series of activities forming a value-chain.[4] The choice of what activities to perform, and subsequent decisions as to how to perform these activities, defines the strategy of the organization. The individual attributes of an organization's strategy interact. For example, the value of an efficient manufacturing process is enhanced when combined with a high-quality sales force. Differing values for K correspond to varying degrees of payoff-interaction among elements of the organization's strategy [38]. As K increases, the difficulty of the task facing strategic decision makers is magnified. Local-search attempts to improve an organization's position on the strategic landscape become ensnared in a web of conflicting constraints.

It is acknowledged that there are limitations to using the NK model as a strategic landscape generator. The model produces a finite graph and presupposes the existence of a strategy space, albeit one which may be poorly understood by strategists. This implies that it is inappropriate to apply the NK model to examine very long run adaptive processes, where organizational fitness is not clearly bounded, and

[4]This activity-based conceptualization has spread beyond studies of strategy to encompass new methods of costing products/services [17].

where the dimensionality of the strategy space itself could change. It is also noted that the NK model assumes a constant value of K for all elements. In reality, the value of K is likely to differ for varying elements of a strategy vector. In the work of [37], a distinction is drawn between *generic activities* which are likely to have an optimal configuration for many firms, for example, the possession of an accounting system. Generic activities (or 'table-stakes'), whilst important for the successful operation of the firm, are usually not strongly interconnected with the non-generic activities of the firm [37]. In contrast, the firm-specific element of strategy are typically highly interconnected, as they embed choices involving trade-offs between alternative strategic configurations [36], [37]. Hence, the NK landscape can be considered to represent these non-generic, interconnected, elements of the strategy vector, rendering the assumption of a constant value of K more plausible.

4.2 The Algorithm

Five characteristics of the problem domain which impact on the design of a simulation model are:

i. the environment is dynamic,

ii. organizations are prone to strategic inertia,

iii. organizations do not knowingly select poorer strategies than the one they already have (election operator),

iv. organizations make errorful *ex-ante* assessments of fitness, and

v. organizations co-evolve.

Each of these factors is embedded in our simulation model. In this study we report results which consider the first three of these factors. Future work will extend this to incorporate the latter two. We note that this model bears passing resemblance to the eleMentals model of [24], which combined a swarm algorithm and an NK landscape, to investigate the development of culture and intelligence in a population of hypothetical beings called eleMentals. However, the strategic model developed in this study is differentiated from the eleMental model, not just on grounds of application domain, but because of the inclusion of an inertia operator, and also by the investigation of both static and dynamic environments.

4.2.1 Dynamic environment

Organizations do not compete in a static environment. Rather they can individually and collectively alter their environment. The environment may also be altered as a result of exogenous events. The second of these factors is implemented in this study by allowing the landscape itself to be respecified. During the course of a simulation run, the strategic landscape can be stochastically subject to minor or major respecification, mimicking a *regime change*, such as the emergence of a new technology, or a change in customer preferences. These respecifications simulate a dynamic environment, and a change in the environment may at least partially negate the value of past learning (adaptation) by organizations.[5] Minor respecifications are simulated by altering the fitness values associated with one of the N dimensions in the NK model, whereas in major changes, the fitness of the entire NK landscape is redefined. The probability that a minor or major respecification occurs is controlled by the modeler.

4.2.2 Inertia

Organizations do not have complete freedom to alter their current strategy. Their adaptive processes are subject to conservatism arising from inertia. Inertia springs from the organization's culture, history, and the mental models of its management [4]. In the simulation strategic inertia is mimicked by implementing a 'strategic anchor'. The degree of inertia can be varied in the simulations from zero to high. In the latter case, the organization is highly constrained from altering its strategic stance. By allowing the weight of this anchor to vary, adaptation processes corresponding to different industries, each with different levels of inertia, can be simulated. Inertia could be incorporated into the PSA in a variety of ways. We have chosen to incorporate it into the velocity update equation, so that the velocity and direction of the particle at each iteration is also a function of the location of its 'strategic anchor'. Therefore for the simulations, equation 1 is altered by adding an additional inertia term

$$v_i(t+1)=v_i(t)+R_1(y_i-x_i(t))+R_2(\hat{y}-x_i(t)+R_3(a_i-x_i(t)) \quad (7)$$

where a_i represents the value of the anchor on dimension i (a full description of the other terms such as R_1 is provided in the pseudo-code below). This anchor can be fixed at the initial position of the particle at the start of the algorithm, or it can be allowed to 'drag', thereby being responsive to the recent adaptive history of the particle. Both the weight attached to the anchor parameter (relative to those attached to pbest and gbest), and in the case of a dragging anchor, the number of periods over which the anchor can drag, can be altered by the modeler.

It is noted that the concept of inertia developed in this paper is not limited to organizations, but is plausibly a general feature of social systems. Hence, the extension of the social swarm model to incorporate inertia may prove useful beyond this study.

4.2.3 Election operator

Real-world organizations do not usually intentionally move to poorer strategies. Hence, an election operator is im-

[5] As noted by [11] (p. xxvii), '*the very processes and values that constitute an organization's capabilities in one context, define its disabilities in another.*'.

plemented, whereby position updates which would worsen an organization's strategic fitness are discarded. In these cases, an organization remains at its current location. One economic interpretation of the election operator, is that strategists carry out a mental simulation or *thought experiment*. If the expected fitness of the proposed strategy appears unattractive, the 'bad idea' is discarded [6], [25]. The simulation therefore incorporates a 'rachet' operator option, which if turned on, ensures that an organization only updates (alters) its strategy if the new strategy being considered is better than its current strategy. By permitting strategists to conduct thought experiment during each iteration of the algorithm, strategists are given a *look-ahead* capability. They can direct their adaptive efforts to the area of the strategic landscape which offer potential.

4.2.4 Outline of algorithm

A number of further modifications to the basic PSA are required. As the strategic landscape is defined using a binary representation, the canonical PSA described above is adapted for the binary case using the *BinPSO* version of the algorithm [23]. The binary version of the PSA is inspired by the idea that an agent's probability of making a binary decision (yes/no, true/false) is a function of both personal and social factors **Eq.** 8.

$$P(\mathbf{x_i(t+1)}=1)=f(\mathbf{x_i(t)},\mathbf{v_i(t)},\mathbf{pbest},\mathbf{gbest},\mathbf{anchor}) \quad (8)$$

The vector $\mathbf{v_i}$ is now interpreted as organization i's predisposition to set each of the n binary strategic choices that they face to one. The higher the value of v_i^j for an individual decision j, the more likely that organization i will choose to set decision $j = 1$, with lower values of v_i^j favoring the choice of decision $j = 0$.

In order to model the tendency of organizations to repeat historically good strategies, values for each dimension of $\mathbf{x_i}$, which match those of **pbest**, should become more probable in the future, and the $Prob(x_i^j = 1)$ should be adjusted towards $pbest_i^j$ on each dimension j. Adding the difference between $pbest_i^j$ and x_i^j for organization i to v_i^j will move the probability thresholds towards 1.0, if the distance is positive ($pbest_i^j = 1$ and $x_i^j = 0$). If the difference between $pbest_i^j$ and x_i^j for organization i is negative ($pbest_i i^j = 0$), and $x_i^j = 1$, adding the difference to v_i^j will move it towards 0.0. The difference in each case is weighted by a random number drawn from U(0,1).[6]

In order to ensure that the vector $\mathbf{v_i(t+1)}$ is mapped into (0,1), a sigmoid transformation is performed on each element j of $\mathbf{v_i(t+1)}$ (**Eq.** 9), and each element of $Sig(\mathbf{v_i(t)})$ is mapped to either 0 or 1 by comparing it with a vector of random numbers $\mathbf{prob_i(t+1)}$ drawn from $U(0,1)$ (**Eq.** 10) .

[6]Similarly, each organization has a tendency to match the values for each dimension of $\mathbf{x_i}$ to those of **gbest**, and its anchor. Therefore, the resulting value of $v_i^j(t+1)$, is influenced by $v_i^j(t)$, and the position of gbest, pbest, and anchor.

$$Sig(v_i^j(t+1))=\frac{1}{1+exp(-v_i^j(t+1))} \quad (9)$$

$$prob_i^j(t+1)<Sig(v_i^j(t+1))\ then\ x_i^j(t+1)=1;\ else\ x_i^j(t+1)=0 \quad (10)$$

The pseudo-code for the algorithm is as follows:

```
For each dimension n
  v[n]=v[n]+R1*(g[n]-x[n])+R2*(p[n]-x[n])+R3*(a[n]-x[n])
  If(v[n]>Max) v[n]=Vmax
  If(v[n]<-Vmax) v[n]=-Vmax
  If(Pr<S(v[n]))t[n]=1
    Else t[n]=0
UpdateAnchor(a)       //if iteratively update anchor
                      //option is selected
```

$R1$, $R2$ and $R3$ are random weights drawn from a uniform distribution ranging from 0 to $R1_{max}$, $R2_{max}$ and $R3_{max}$ respectively, and they weight the importance attached to the gbest, pbest and anchor in each iteration of the algorithm. $R1_{max}$, $R2_{max}$ and $R3_{max}$ are constrained to sum up to 4.0. x is the particle's actual position, g is the global best position, p each particle's personal best position and a is the position of the particle's anchor. V_{max} is set to 4.0. Pr is a probability value drawn from a uniform distribution ranging from 0 to 1, and S is the sigmoid function: $S(x) = \frac{1}{1+exp(-x)}$, which squashes v into a 0 to 1 range. t is a temporary record which is used in order to implement conditional moving. If the new strategy is accepted, t is copied into x, otherwise t is discarded and x remains unchanged.

5 Results

This section provides the results from our simulation study. As the adaptive process is stochastic, and as the initialization of the position and velocity for each organization is random, each simulation run describes a single sample-path through time. There are many possible sample-paths, so the results of the simulations are averaged over multiple (30) runs in an attempt to uncover prevalent characteristics of the sample paths which the system can give rise to. All simulations were run for 5,000 iterations, and all reported fitnesses are the average population fitnesses, and average environment best fitnesses, across 30 separate simulation runs. On each of the simulation runs, the NK landscape is specified anew, and the positions and velocities of particles are randomly initialized at the start of each run. A population of 20 particles is employed, with a neighborhood of size 18. The choice of a high value for the neighborhood, relative to the size of the population, arises from the observation that real-world organizations know the profitability of their competitors.

Tables (1, 2 and 3) provide the results for each of fourteen distinct PSA variants, at the end of 5,000 iterations, across a number of static and dynamic NK landscape scenarios. In each scenario, the same series of simulations are undertaken. Initially, a basic PSA is employed, without an anchor or a rachet (conditional move) operator. This

simulates a population of organizations searching a strategic landscape, where the population has no strategic inertia, and where organizations do not utilize a rachet operator in deciding whether to alter their position on the strategic landscape.

The basic PSA is then supplemented by a series of strategic anchor formulations, ranging from an anchor which does not change position during the simulation (initial anchor) to one which can adapt after a time-lag (moving anchor). Two lag periods are examined, a 20 and a 50 iteration lag. Differing weights can be attached to the inertia term in the velocity equation, ranging from 0 (inertia is turned off) to a maximum of 4. To determine whether the weight factor has a critical impact on the results, results are reported for weight values of both 1 and 3. Next, to isolate the effect of the rachet, the conditional move operator is implemented, and inertia is turned off. Finally, to ascertain the dual effect of both rachet and inertia when they are combined, the inertia simulations outlined above are repeated with the rachet operator turned on.

Real world strategy vectors consist of a large array of strategic decisions. A value of N=96 was chosen in defining the landscapes in this simulation. It is noted that there is no unique value of N that could have been selected, but the selection of very large values are not feasible due to computational limitations. However, a binary string of 96 bits provides 2^{96}, or approximately 10^{28}, distinct choices of strategy. It is also noted that we would expect the dimensionality of the strategy vector to exceed the number of organizations in the population, hence the size of the population is kept below 96, and a value of 20 is chosen. A series of landscapes of differing K values (0,4 and 10), representing differing degrees of fitness inter-connectivity, were used in the simulations.

5.1 Static Landscape

Table 1 and figures 2 and 3, provide the results for a static NK landscape.[7] Examining these results suggests that the basic PSA, without inertia or rachet operators, performs poorly, even on a static landscape. The average of the average batch populational fitnesses obtained after 5,000 iterations is not better than random search (the expected value of a random point on the landscape is 0.50), suggesting that unfettered adaptive efforts, based on communication between organizations (gbest), and a memory of good past strategies (pbest) is not sufficient to achieve high levels of populational fitness. When a series of anchor mechanisms simulating strategic inertia are added to the basic PSA, the results are not qualitatively altered from those of the basic PSA. This suggests that communication and inertia alone, are not sufficient for the attainment of high levels of populational strategic fitness.

[7]These simulations were also undertaken with a neighborhood size of four, to determine whether the results were sensitive to neighborhood size. No significant differences in the results between the two neighborhood sizes was noted. As a result, the remaining simulations were run with a neighborhood of size 18.

When a rachet operator is added to the basic PSA, a significant improvement in both average populational, and average environment best fitness is obtained across landscapes of all K values, suggesting that the simple decision heuristic of *only abandon a current strategy for a better one* can lead to notable increases in populational fitness. Finally, the results of a series of combination anchor and rachet mechanisms are reported. Virtually all of these combinations lead to significantly (at the 5% level) enhanced levels of populational fitness (against the rachet-only PSA), suggesting that inertia can be beneficial, when combined with a rachet mechanism. Examining the combined rachet and anchor results in more detail, the best results are obtained when the anchor is not fixed at the initial location of each particle on the landscape, but when it is allowed to 'drag' or adapt, over time. It is also noted that the results are not qualitatively sensitive to the weight value (1 or 3).

5.2 Dynamic Landscape

The real world is rarely static, and changes in the environment can trigger adaptive behavior by agents in a system [3]. In this simulation, the landscape can change at a variety of time scales, and the size of the relocation 'jump' of the optimum position on the landscape can vary. Therefore, the environment can be changed with varying temporal, and spatial severity [3]. Two specific scenarios are examined. Table 2 and figures 4 and 5, provides the results for the case where a single dimension of the NK landscape is respecified in each iteration of the algorithm with a probability of P=0.00025. Table 3 and figures 6 and 7, provides the results for the case where the entire NK landscape is respecified with the same probability. When the landscape is wholly or partially respecified, the benefits of past strategic learning by organizations is eroded.

Qualitatively, the results in both scenarios are similar to those obtained on the static landscape. The basic PSA, even if supplemented by an anchor mechanism, does not perform any better than random search. Supplementing the basic PSA with the rachet mechanism leads to a significant improvement in populational fitness, with a further improvement in fitness occurring when the rachet is combined with an anchor. In the latter case, an adaptive or dragging anchor gives better results than a fixed anchor, but the results between differing forms of dragging anchor do not show a clear dominance for any particular form. As for the static landscape case, the results for the combined rachet / anchor, are relatively insensitive to the weight value (1 or 3).

6 Conclusions

The objective of this study has been to examine the impact of strategic inertia on the dynamic adaptation of a population of organizations. A novel synthesis of a strategic landscape defined using the NK model, and a Particle Swarm metaphor to model the adaption of organizations

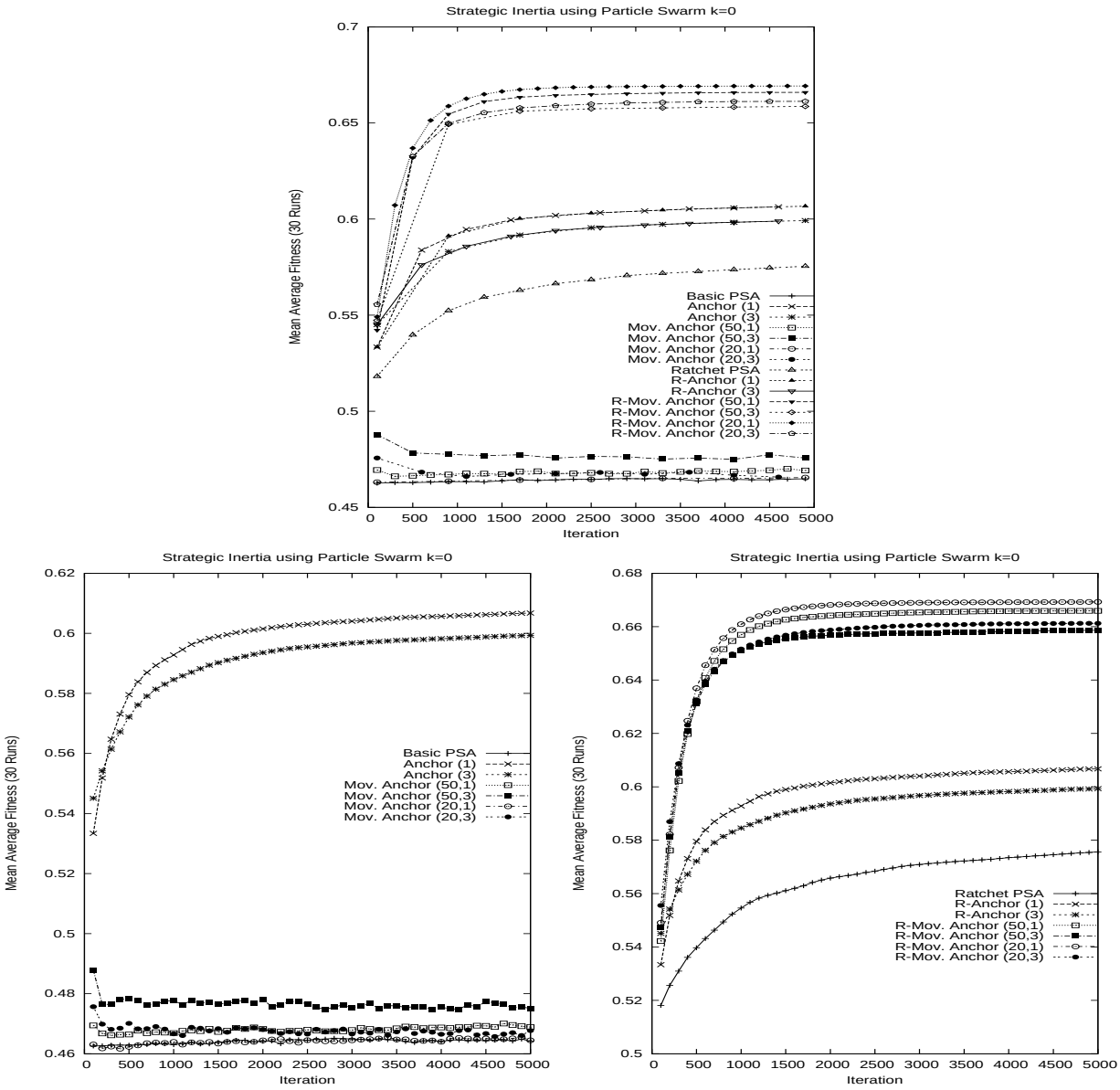


Figure 2: Plot of the mean average fitness on the static landscape where k=0.

on this landscape, is used to construct a simulation model. Adoption of the swarm metaphor allows the incorporation of both social and individual learning mechanisms, and the basic algorithm can be easily adapted to include other search heuristics such as election and inertia.

The results suggest that a degree of strategic inertia, in the presence of an election operator, can assist rather than hamper the adaptive efforts of populations of organizations in static and slowly changing strategic environments. The results also provide an interesting perspective on the claim by [13] that inertia may be a consequence of market-selection processes. The results indicate that there may be good reasons, from a populational perspective, for market selection processes to encourage populations of organizations which exhibit a degree of inertia. Despite the claim for the importance of social learning in populations, the results suggest that social learning alone is of limited benefit, unless supported by an election mechanism.

In the construction of any simulation model, aspects of the real-world system of interest must be omitted. In this study, we omit the cost of making a strategic adjustment,[8] and we omit an explicit birth-death process for the population of organizations.[9] We note that the effect of the gbest, pbest and inertia anchors, is to pin each organization on the landscape. To the extent that the entire collection of organizations have converged to a relatively small region of the landscape, they may find it impossible to migrate

[8] Although we note that incorporating such costs would likely enhance the value of inertia.

[9] It could be argued that although there is no explicit selection process, the effect of including a gbest term in the model is to incorporate an implicit form of selection, in that organizations with poor strategies are drawn towards the location of gbest, mimicking a selection process.

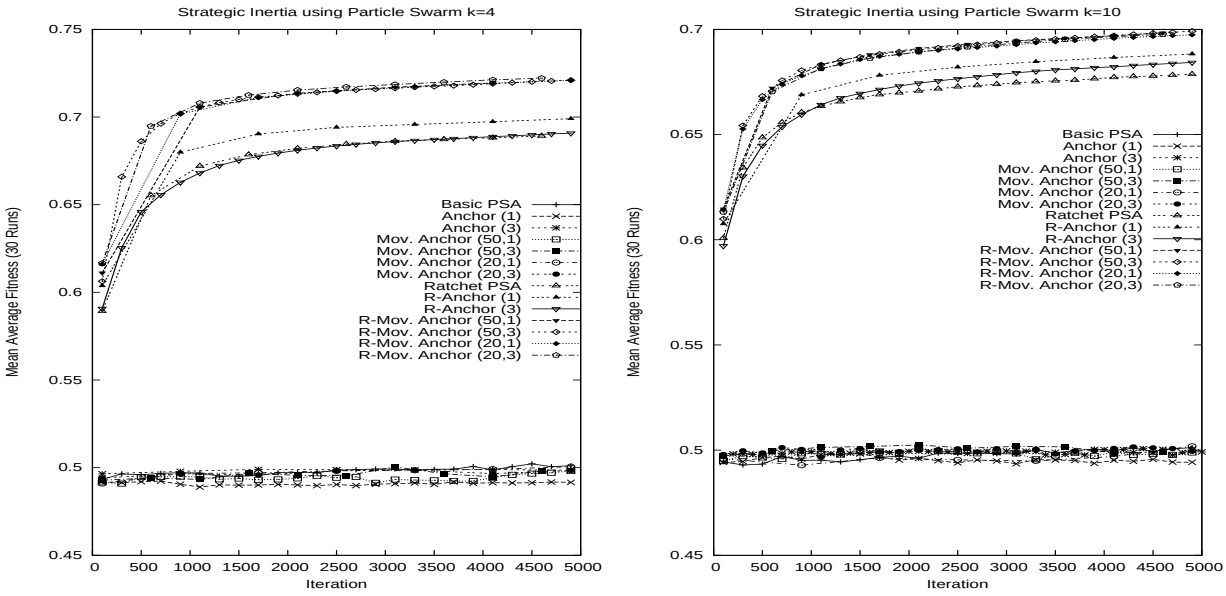


Figure 3: Plot of the mean average fitness on the static landscape where k=4 (left), and where k=10 (right).

to a new high-fitness region of the landscape if that region moves far away from their current location. In real-world environments, this is compensated for by the birth of new organizations.

This study describes the *OrgSwarm* simulator, and reports the results of initial simulations using this model. Future work will extend the range of strategic scenarios, and parameter settings considered. In particular we intend to examine the process of strategic adaptation when strategists make errorful assessments of the fitness of proposed strategies. We also intend to incorporate a co-evolutionary aspect into the model (mimicking direct competition between organizations), wherein the fitness of a strategy is partially determined by the number of organizations which are pursing similar strategies.

References

[1] Abido, M. (2002). Optimal power flow using particle swarm optimization, *Electrical power & Energy Systems*, 24:563-571.

[2] Barnett, W. and Hansen, M. (1996). The Red Queen in Organizational Evolution, *Strategic Management Journal*, 17:139-157.

[3] Blackwell, T. (2003). Swarms in Dynamic Environments, *in Proceedings of GECCO 2003*, Lecture Notes in Computer Science (2723), Springer-Verlag, Berlin, pp. 1-12.

[4] Boeker, W. (1989). Strategic Change: The Effects of Founding and History, *Academy of Management Journal*, 32(3):489-515.

[5] Bonabeau, E., Dorigo, M. and Theraulaz, G. (1999). *Swarm Intelligence: From natural to artificial systems*, Oxford: Oxford University Press.

[6] Birchenhall, C. (1995). Modular Technical Change and Genetic Algorithms, *Computational Economics*, 8:233-253.

[7] Brabazon, A., Silva, A., Ferra de Sousa, T., O'Neill, M. and Matthews, R. (2004). A Particle Swarm Model of Organizational Adaptation, *in Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2004), Lecture Notes in Computer Science (3102)*, Deb et. al. Eds., Seattle, USA, June 26-30, 2004, 1:12-23, Berlin: Springer-Verlag.

[8] Brabazon, A., Silva, A., Ferra de Sousa, T., O'Neill, M. and Matthews, R. (2004). Investigating Organizational Strategic Inertia Using a Particle Swarm Model, *in Proceedings of the 2004 IEEE Congress on Evolutionary Computation*, 1:652-659, IEEE Press: New Jersey.

[9] Carroll, G. and Hannan, T. (1995). *Organizations in Industry: strategy, structure and selection*, New York: Oxford University Press.

[10] Child, J. (1972). Organizational Structure, Environment and Performance: The Role of Strategic Choice, *Sociology*, 6:2-22.

[11] Christensen, C. (1997). *The Innovator's Dilemma*, (HarperBusiness Essentials, 2003 edition), New York: HarperBusiness Essentials.

[12] Hannan, M. and Freeman, J. (1977). The Populational Ecology of Organizations, *American Journal of Sociology*, 82(5): 929-964.

[13] Hannan, M. and Freeman, J. (1984). Structural Inertia and Organizational Change, *American Sociological Review*, 49:149-164.

[14] Helfat, C. (1994). Evolutionary Trajectories in Petroleum Firm R&D, *Management Science*, 40(12):1720-1747.

[15] Gavetti, G. and Levinthal, D. (2000). Looking Forward and Looking Backward: Cognitive and Experiential Search, *Administrative Science Quarterly*, 45:113- 137.

[16] Hammer, M. and Champy, J. (2001). Reengineering the corporation (revised edition): A manifesto for business revolution, HarperBusiness: New York.

[17] Kaplan, R. and Cooper, R. (1998). *Cost and effect: Using integrated cost systems to drive profitability and performance*, Boston, Massachusetts: Harvard Business School Press.

[18] Kauffman, S. and Levin, S. (1987). Towards a General Theory of Adaptive Walks on Rugged Landscapes, *Journal of Theoretical Biology*, 128:11-45.

[19] Kauffman, S. (1993). *The Origins of Order*, Oxford,England: Oxford University Press.

[20] Kauffman, S., Lobo, J. and MacReady, W. (1998). Optimal Search on a Technology Landscape, *Santa Fe Institute Working Paper 98-10-091*.

[21] Kennedy, J. and Eberhart, R. (1995). Particle swarm optimization, *Proceedings of the IEEE International Conference on Neural Networks*, December 1995, pp. 1942-1948.

[22] Kennedy, J. (1997). The particle swarm: Social adaptation of knowledge, *in Proceedings of the International Conference on Evolutionary Computation*, pp. 303-308, Piscataway, New Jersey:IEEE Press.

[23] Kennedy, J. and Eberhart, R. (1997). A discrete binary version of the particle swarm algorithm, *Proceedings of the Conference on Systems, Man and Cybernetics*, pp. 4104-4109, Piscataway, New Jersey: IEEE Press.

[24] Kennedy, J. (1999). Minds and Cultures: Particle Swam Implications for Beings in Sociocognitive Space, *Adaptive Behavior*, 7(3/4):269-288.

[25] Kennedy, J., Eberhart, R. and Shi, Y. (2001). *Swarm Intelligence*, San Mateo, California: Morgan Kauffman.

[26] Kitts, B., Edvinsson, L. and Beding, T. (2001). Intellectual capital: from intangible assets to fitness landscapes, *Expert Systems with Applications*, 20:35-50.

[27] Levinthal, D. (1991). Random Walks and Organisational Mortality, *Administrative Science Quarterly*, 36:397-420.

[28] Levinthal, D. (1997). Adaptation on Rugged Landscapes, *Management Science*, 43(7):934-950.

[29] Lobo, J. and MacReady, W. (1999). Landscapes: A Natural Extension of Search Theory, *Santa Fe Institute Working Paper 99-05-037*.

[30] Makadok, R. and Walker, G. (1996). Search and Selection in the Money Market Fund Industry, *Strategic Management Journal*, 17:39-54.

[31] March, J. (1981). Footnotes to Organizational Change, *Administrative Science Quarterly*, 26:563-577.

[32] March, J. (1991). Exploration and Exploitation in Organisational Learning, *Organization Science*, 2(1):71-87 .

[33] Nelson, R. and Winter, S. (1982). *An Evolutionary Theory of Economic Change*, Cambridge, Massachusetts, Harvard University Press.

[34] Ourique, C., Biscaia, E. and Pinto, J. (2002). The use of particale swarm optimization for dynamical analysis in chemical processes, *Computers and Chemical Engineering*, 26:1783-1793.

[35] Porter, M. (1985). *Competitive Advantage: Creating and Sustaining Superior Performance*, New York, The Free Press.

[36] Porter, M. (1996). What is Strategy?, *Harvard Business Review*, Nov-Dec, 61-78.

[37] Porter, M. and Siggelkow, N. (2001). Contextuality within Activity Systems,*Harvard Business School Working Paper Series*, No. 01-053, 2001.

[38] Rivkin, J. (2000). Imitation of Complex Strategies, *Management Science*, 46(6):824-844.

[39] Silva, A., Neves, A. and Costa, E. (2002). An empirical comparision of particle swarm and predator prey optimisation, in *Lecture Notes in Artificial Intelligence (2464) - Proceedings of AICS 2002*, edited by M. O'Neill, R.F.E. Sutcliffe, C. Ryan, M. Eaton, N.J.L. Griffith Springer-Verlag, Berlin, pp. 103-110.

[40] Simon, H. (1993). Strategy and Organizational Evolution, *Strategic Management Journal*, 14:131-142.

[41] Strumsky, D. and Lobo, J. (2002). If it isn't broken, don't fix it: Extremal search on a technology landscape, *Santa Fe Institute Working Paper 03-02-003*.

[42] Stuart, T. and Podolny, J. (1996). Local Search and the Evolution of Technological Capabilities, *Strategic Management Journal*, 17:21-38.

[43] Sull, D. (1999). Why Good Companies Go Bad, *Harvard Business Review*, 77(4):42-52.

[44] Tanev, I. and Shimohara, K. (2003). On Role of Implicit Interaction and Explicit Communications in Emergence of Social Behaviour in Continuous Predators-Prey Pursuit Problem, *in Proceedings of GECCO 2003, Lecture Notes in Computer Science (2723)*, Springer-Verlag, Berlin, pp. 74-85.

[45] Tushman, M. and O'Reilly, C. (1996). Ambidextrous Organizations: Managing Evolutionary and Revolutionary Change, *California Management Review*, 38(4):8-30.

[46] Wright, S. (1932). The roles of mutation, inbreeding, crossbreeding and selection in evolution, *Proceedings of the Sixth International Congress on Genetics*, 1:356-366.

Algorithm	**Fitness**		
	(N=96, K=0)	**(N=96, K=4)**	**(N=96, K=10)**
Basic PSA	0.4641 (0.5457)	0.5002 (0.6000)	0.4991 (0.6143)
Initial Anchor, w=1	0.4699 (0.5484)	0.4921 (0.5967)	0.4956 (0.6102)
Initial Anchor, w=3	0.4943 (0.5591)	0.4994 (0.5979)	0.4991 (0.6103)
Mov. Anchor (50,1)	0.4688 (0.5500)	0.4960 (0.6003)	0.4983 (0.6145)
Mov. Anchor (50,3)	0.4750 (0.5631)	0.4962 (0.6122)	0.5003 (0.6215)
Mov. Anchor (20,1)	0.4644 (0.5475)	0.4986 (0.6018)	0.5001 (0.6120)
Mov. Anchor (20,3)	0.4677 (0.5492)	0.4994 (0.6156)	0.4994 (0.6229)
Rachet PSA	0.5756 (0.6021)	0.6896 (0.7143)	0.6789 (0.7035)
R-Initial Anchor, w=1	0.6067 (0.6416)	0.6991 (0.7261)	0.6884 (0.7167)
R-Initial Anchor, w=3	0.5993 (0.6361)	0.6910 (0.7213)	0.6844 (0.7099)
R-Mov. Anchor (50,1)	0.6659 (0.6659)	0.7213 (0.7456)	0.6990 (0.7256)
R-Mov. Anchor (50,3)	0.6586 (0.6601)	0.7211 (0.7469)	0.6992 (0.7270)
R-Mov. Anchor (20,1)	0.6692 (0.6695)	0.7211 (0.7441)	0.6976 (0.7243)
R-Mov. Anchor (20,3)	0.6612 (0.6627)	0.7228 (0.7462)	0.6984 (0.7251)

Table 1: Average (environment best) fitnesses after 5,000 iterations, static landscape.

Algorithm	**Fitness**		
	(N=96, K=0)	**(N=96, K=4)**	**(N=96, K=10)**
Basic PSA	0.4667 (0.5245)	0.4987 (0.5915)	0.4955 (0.6065)
Initial Anchor, w=1	0.4658 (0.5293)	0.4908 (0.5840)	0.4957 (0.6038)
Initial Anchor, w=3	0.4922 (0.5513)	0.4992 (0.5953)	0.5001 (0.60852)
Mov. Anchor (50,1)	0.4614 (0.5200)	0.4975 (0.5927)	0.5008 (0.6044)
Mov. Anchor (50,3)	0.4691 (0.5400)	0.4975 (0.6040)	0.4987 (0.6174)
Mov. Anchor (20,1)	0.4686 (0.5315)	0.5010 (0.6002)	0.4958 (0.6099)
Mov. Anchor (20,3)	0.4661(0.5434)	0.4964(0.6084)	0.4988 (0.6137)
Rachet PSA	0.5783 (0.6056)	0.6859 (0.7096)	0.6808 (0.7066)
R-Initial Anchor, w=1	0.6207 (0.6553)	0.6994 (0.7330)	0.6895 (0.7142)
R-Initial Anchor, w=3	0.5927 (0.6239)	0.6900 (0.7182)	0.6850 (0.7140)
R-Mov. Anchor (50,1)	0.6676 (0.6688)	0.7187 (0.7438)	0.6987 (0.7241)
R-Mov. Anchor (50,3)	0.6696 (0.6696)	0.7203 (0.7462)	0.6989 (0.7264)
R-Mov. Anchor (20,1)	0.6689 (0.6694)	0.7193 (0.7426)	0.6974 (0.7251)
R-Mov. Anchor (20,3)	0.6594 (0.6622)	0.7221 (0.7450)	0.6987 (0.7280)

Table 2: Average (environment best) fitnesses after 5,000 iterations, 1 dimension respecified periodically.

Algorithm	**Fitness**		
	(N=96, K=0)	**(N=96, K=4)**	**(N=96, K=10)**
Basic PSA	0.4761 (0.5428)	0.4886 (0.5891)	0.4961 (0.6019)
Initial Anchor, w=1	0.4819 (0.5524)	0.4883 (0.5822)	0.4982 (0.6075)
Initial Anchor, w=3	0.5021 (0.5623)	0.4967 (0.5931)	0.4998 (0.6047)
Mov. Anchor (50,1)	0.4705 (0.5450)	0.4894 (0.5863)	0.4974 (0.6008)
Mov. Anchor (50,3)	0.4800 (0.5612)	0.4966 (0.6053)	0.5010 (0.6187)
Mov. Anchor (20,1)	0.4757 (0.5520)	0.4926 (0.5867)	0.4985 (0.6097)
Mov. Anchor (20,3)	0.4824 (0.5632)	0.4986 (0.6041)	0.5004 (0.6163)
Rachet PSA	0.5877 (0.6131)	0.6802 (0.7092)	0.6754 (0.7015)
R-Initial Anchor, w=1	0.6187 (0.6508)	0.6874 (0.7180)	0.6764 (0.7070)
R-Initial Anchor, w=3	0.6075 (0.6377)	0.6841 (0.7130)	0.6738 (0.7017)
R-Mov. Anchor (50,1)	0.6517 (0.6561)	0.7134 (0.7387)	0.6840 (0.7141)
R-Mov. Anchor (50,3)	0.6597 (0.6637)	0.7049 (0.7304)	0.6925 (0.7225)
R-Mov. Anchor (20,1)	0.6575 (0.6593)	0.7152 (0.7419)	0.6819 (0.7094)
R-Mov. Anchor (20,3)	0.6689 (0.6700)	0.7158 (0.7429)	0.6860 (0.7147)

Table 3: Average (environment best)fitnesses after 5,000 iterations, entire landscape respecified periodically.

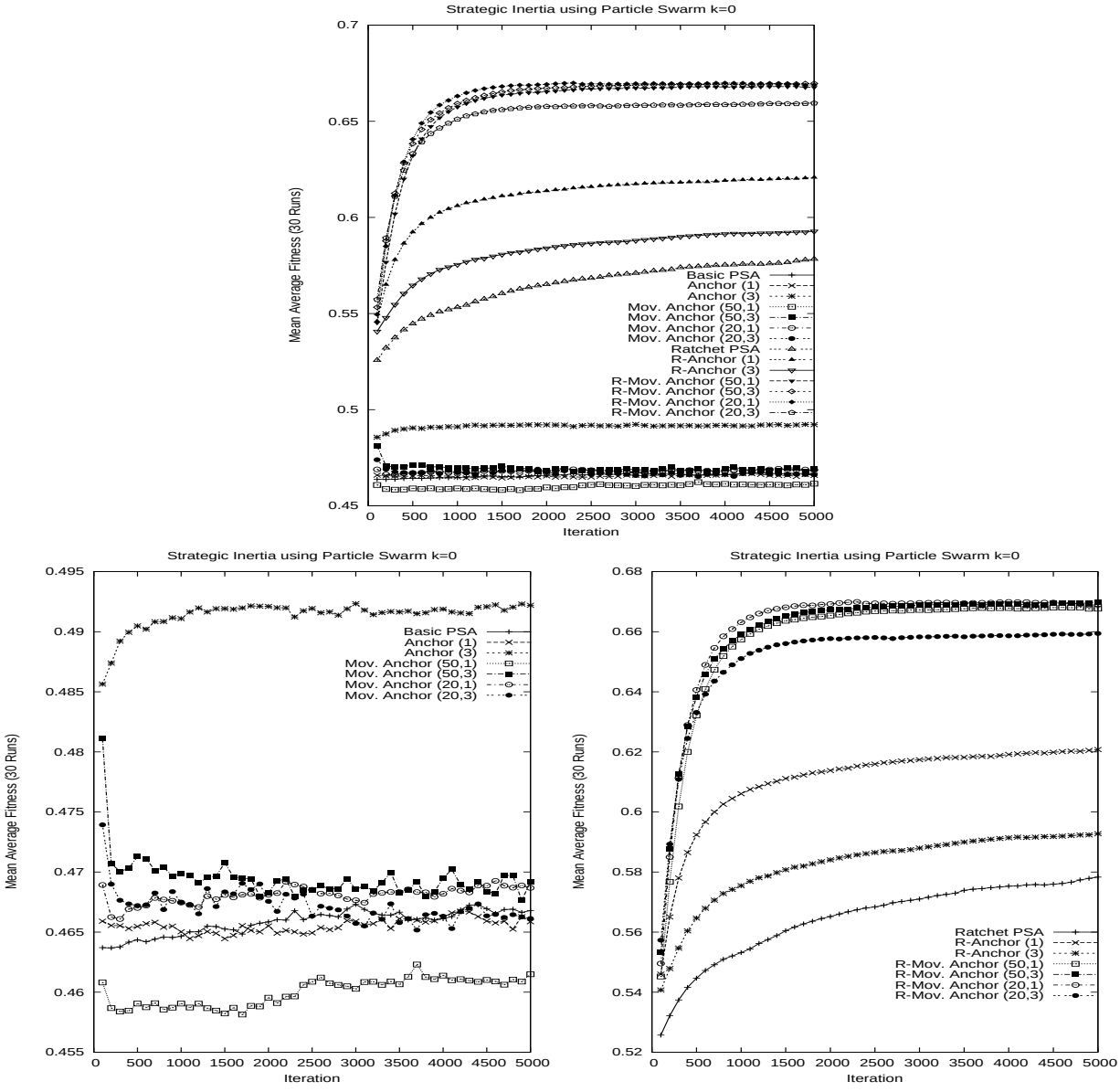


Figure 4: Plot of the mean average fitness on the dynamic landscape (one dimension of the landscape is respecified periodically) where k=0.

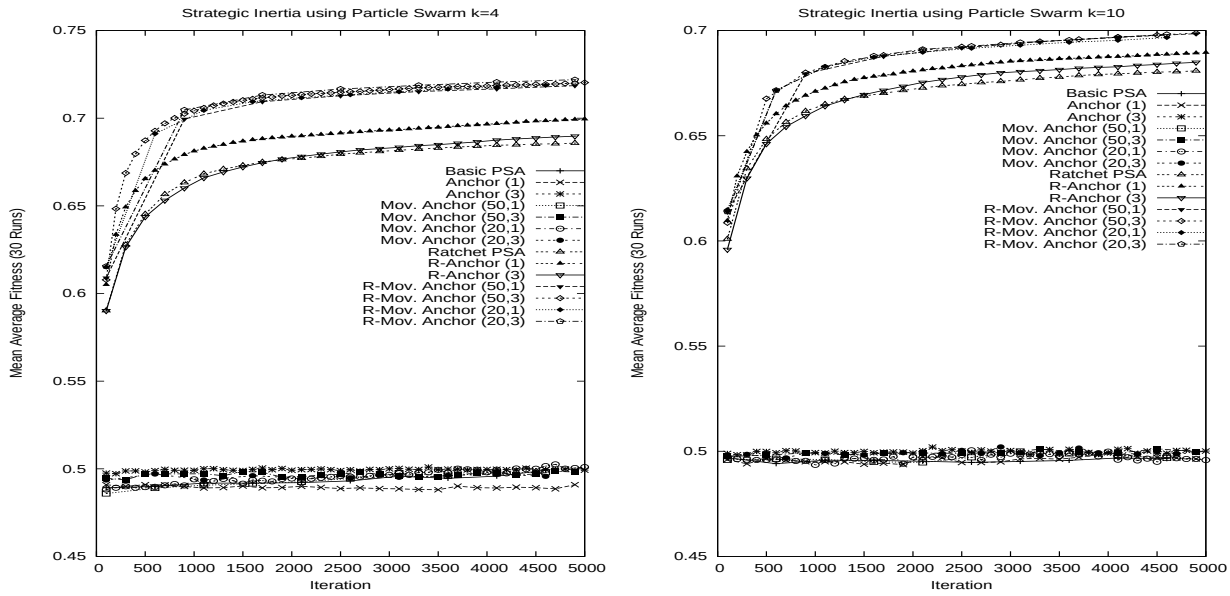


Figure 5: Plot of the mean average fitness on the dynamic landscape (one dimension of the landscape is respecified periodically) where k=4 (left), and where k=10 (right).

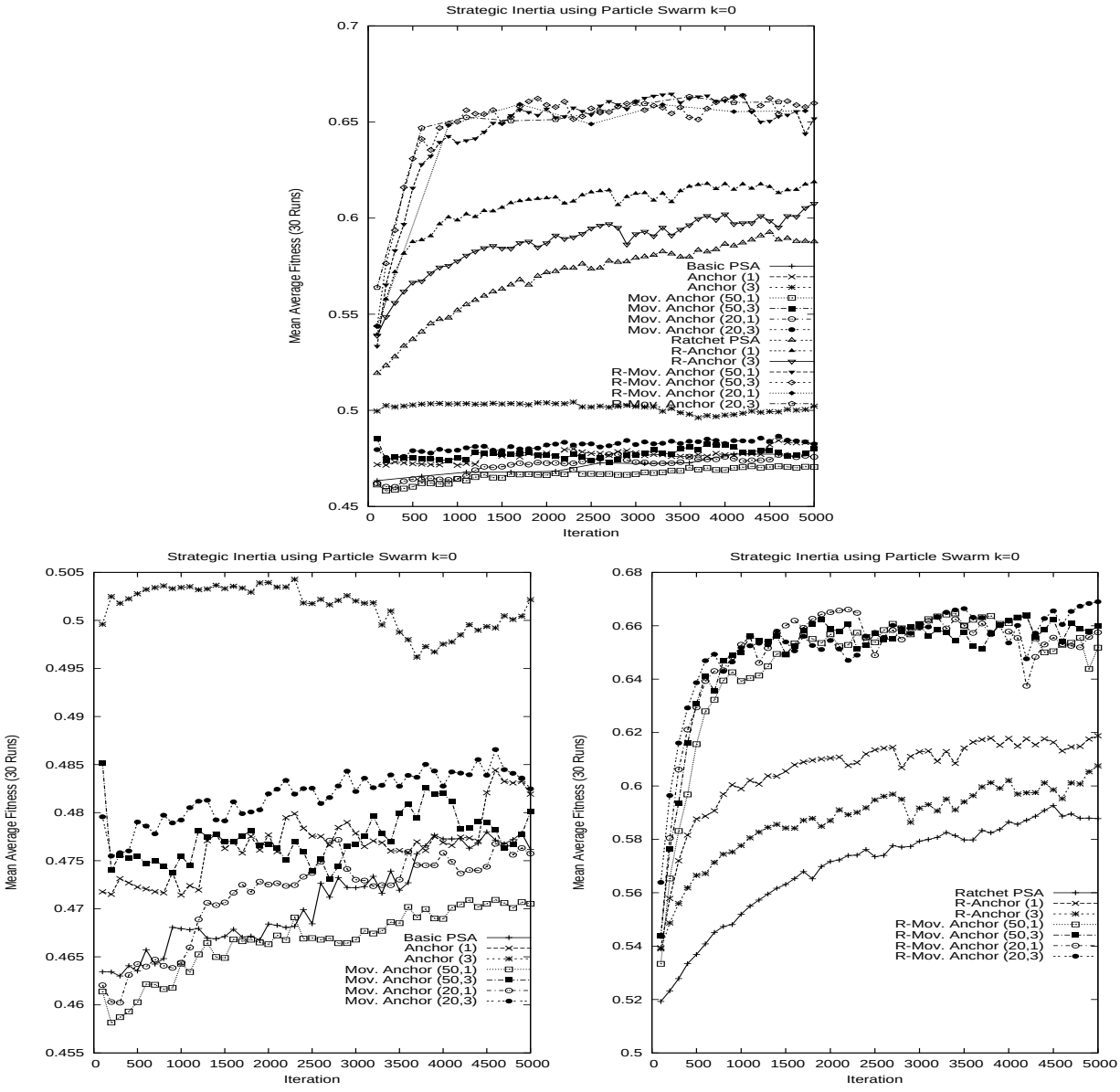


Figure 6: Plot of the mean average fitness on the dynamic landscape (entire landscape respecified periodically) where k=0.

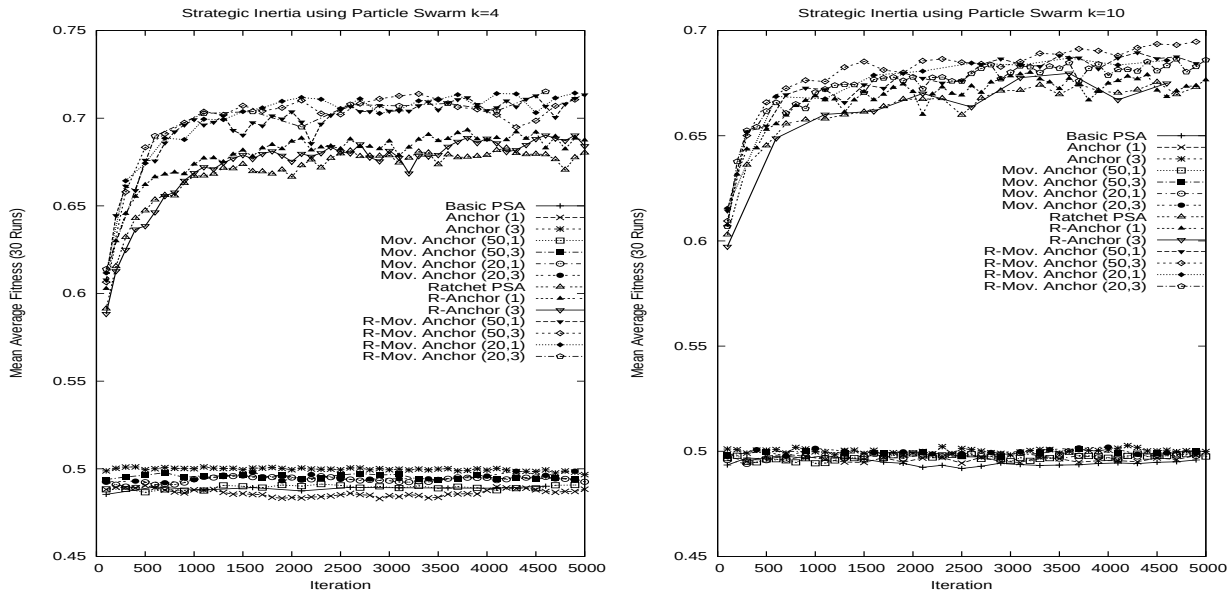


Figure 7: Plot of the mean average fitness on the dynamic landscape (entire landscape respecified periodically) where k=4 (left), and where k=10 (right).

Towards Improving Clustering Ants: An Adaptive Ant Clustering Algorithm

André L. Vizine[1,2], Leandro N. de Castro[1,2], Eduardo R. Hruschka[1], Ricardo R. Gudwin[2]

[1]Catholic University of Santos (UniSantos)
R. Carvalho de Mendonça, 144, 11070-906, Santos/SP, Brasil
{vizine,lnunes,erh}@unisantos.br

[2]State University of Campinas (Unicamp)
DCA–FEEC–UNICAMP, Cx. Postal 6101, 13083-852, Campinas /SP, Brazil.
gudwin@dca.fee.unicamp.br

Keywords: Ant clustering algorithm, Data clustering, Visual data mining

Received: July 15, 2004

Among the many bio-inspired techniques, ant-based clustering algorithms have received special attention from the community over the past few years for two main reasons. First, they are particularly suitable to perform exploratory data analysis and, second, they still require much investigation to improve performance, stability, convergence, and other key features that would make such algorithms mature tools for diverse applications. Under this perspective, this paper proposes both a progressive vision scheme and pheromone heuristics for the standard ant-clustering algorithm, together with a cooling schedule that improves its convergence properties. The proposed algorithm is evaluated in a number of well-known benchmark data sets, as well as in a real-world bioinformatics dataset. The achieved results are compared to those obtained by the standard ant clustering algorithm, showing that significant improvements are obtained by means of the proposed modifications. As an additional contribution, this work also provides a brief review of ant-based clustering algorithms.

Povzetek: Članek opisuje izboljšan algoritem grupiranja na osnovi pristopa kolonij mravelj.

1 Introduction

Over the past few years, several different types of biologically inspired algorithms have been proposed in the literature (Paton, 1994; de Castro & Von Zuben, 2004). Among these, some have obtained special attention from the scientific community, such as those based on swarm systems (Bonabeau et al., 1999; Kennedy et al., 2001), which are inspired by the social behavior of living organisms. This relatively new field of investigation has originated different types of algorithms for the solution of complex problems in many different domains. Under this perspective, the problems usually tackled involve search, optimization, and data analysis tasks. The main reasons by which swarm based approaches are useful for solving such problems are (Bonabeau et al., 1999; Kennedy et al., 2001): (i) they require little information about the problem at hand (e.g. in clustering problems a data set to be grouped); and (ii) they usually can perform both broad and parallel searches over the space of potential solutions by means of a population (swarm) of candidate solutions.

Despite the broad usefulness of current bio-inspired algorithms, most of them can be further improved, mainly to enhance performance and applicability. In this sense, this work focuses on ant-based clustering algorithms, whose main underlying concepts are based on the way real ants clean their nests and organize dead bodies in their colonies. Considering a more practical computational perspective, these algorithms are basically designed by considering the concept of a 2D grid where objects (data) are laid at random and then automatically organized. A set of ant-like agents is allowed to move throughout the grid, picking up and dropping objects (data) based on their similarity degree within a certain neighborhood.

One difficulty in applying ant-clustering algorithms to solve complex problems comes from the fact that, in most cases, they generate a number of clusters that is much larger than the *natural* number of clusters. Furthermore, these algorithms usually do not stabilize in a particular clustering solution; that is, they constantly construct and deconstruct clusters during the iterative procedure of adaptation. In order to overcome the aforementioned difficulties and, consequently, improve the quality of the results obtained, we propose an Adaptive Ant-Clustering Algorithm (A^2CA), which is more robust in terms of the number of clusters found and tends to converge into good solutions while the clustering process

evolves. To achieve these goals, three main modifications are introduced in the standard ant-clustering algorithm proposed by Lumer and Faieta (1994): (i) a cooling schedule for the parameter that controls the probability of ants picking up objects from the grid; (ii) a progressive vision field that allows ants to 'see' over a wider area; and (iii) the use of a pheromone function added to the grid as a way to promote reinforcement for the dropping of objects at more dense regions of the grid. These modifications favor an adaptive clustering process, in the sense that the proposed algorithm tends to converge to stable clusters. In addition to the contributions to the algorithm itself, this paper also brings a brief historical review of ant-based clustering algorithms, emphasizing their main features when compared with the standard ant-clustering algorithm proposed by Lumer and Faieta (1994).

The paper is organized as follows. Section 2 provides a brief review of the standard ant-clustering algorithm (Lumer & Faieta, 1994), which, for the sake of brevity, is referred to as SACA in this work. In Section 3, we present our proposed algorithm (A^2CA), which, in Section 4 is experimentally compared to the SACA in three synthetic and one real-world dataset. Section 5 provides a brief survey of related works, whereas Section 6 concludes the paper and points out some avenues for future work.

2 Standard Ant Clustering Algorithm: SACA

The Standard Ant Clustering Algorithm (SACA), introduced by Lumer and Faieta (1994), assumes that ants perform random walks on a two-dimensional grid on which objects (data) are laid down at random. Independently of the dimension of the input data, each datum is randomly projected onto a cell of the grid. A grid cell (or patch) is thus responsible for hosting the index of a specific input pattern, indicating the relative position of the datum in the two-dimensional grid. The general idea is to have items, which are similar in their original N-dimensional space, in neighboring regions of the grid. In other words, data indices that are neighbors in the grid indicate patterns that are similar in their original space of attributes. In this context, it is assumed that each site or cell on the grid can be occupied by at most one object, and one of the two following situations may occur: (i) one ant holds an object i and evaluates the probability of dropping it in its current position; (ii) an ant is unloaded and evaluates the probability of picking up an object. At each discrete time step, an ant is selected at random and can either pick up or drop an object at its current location.

The probability of picking up an object increases with low-density neighborhoods and decreases with high similarity among objects in the surrounding area. The probability of dropping an object, by contrast, increases with high densities of similar objects in the neighborhood. More specifically, assume that $d(i,j)$ is the Euclidean distance between objects i and j in their N-dimensional space. The density dependent function for object i, at a particular grid location, is defined by the following expression:

$$f(i) = \begin{cases} \frac{1}{s^2}\sum_j (1 - d(i,j)/\alpha) & \text{if } f(i) > 0 \\ 0 & \text{otherwise.} \end{cases}, \quad (1)$$

where s^2 is the number of cells in the surrounding area of i, and α is a constant that scales the dissimilarities among objects. The maximum value for $f(i)$ is obtained if, and only if, all the sites in the neighborhood are occupied by equal objects. Assuming the density dependent function presented in Eq. (1), the probability of picking up and dropping an object i is given by Eqs. (2) and (3), respectively:

$$P_{pick}(i) = \left(\frac{k_p}{k_p + f(i)} \right)^2, \quad (2)$$

$$P_{drop}(i) = \begin{cases} 2f(i) & \text{if } f(i) < k_d; \\ 1 & \text{otherwise.} \end{cases}, \quad (3)$$

where the parameters k_p and k_d are threshold constants equal to 0.1 and 0.15, respectively. Note that $f(i) \in [0,1]$. Thus, if $f(i) << k_p$, then $P_{pick} \approx 1$, leading to high probabilities of picking up objects in low density regions. Similarly, $P_{pick} \approx 0$ if $f(i) >> k_p$, meaning that objects are unlikely to be removed from dense regions. In the case of P_{drop}, it is also possible to observe that if $f(i) << k_d$, $P_{drop} \approx 0$, whereas if $f(i) \geq k_d$ the ant drops the object.

Whenever a loaded ant decides to drop the object it is carrying, it looks for the first empty cell in its vicinity in which to do so (its current position can be already occupied by another object). A time step finishes with the selected ant moving to one of its four adjacent nodes, each direction of motion being equally likely.

3 Adaptive Ant Clustering Algorithm: A^2CA

The Adaptive Ant Clustering Algorithm (A^2CA) was developed by taking further inspiration from biological systems. In particular, A^2CA was inspired by the fact that termites, while building their nests, deposit pheromone on soil pellets and this serves as a reinforcement signal to other termites placing more pellets on the same region of the space (Camazine et al., 2001). Another biological observation taken into account while developing A^2CA was the fact that ants can sense not only its immediate neighborhood environment, but a broader range that may vary from ant to ant and with time. Therefore, A^2CA has two main modifications in relation to SACA: (i) a progressive vision scheme, and (ii) the inclusion of pheromone on the grid cells. In addition, we adopt a cooling schedule for the parameter that drives the picking probability (k_p).

3.1 Cooling Schedule for k_p

In addition to the modifications that led to the development of A^2CA, one simple modification was previously introduced in SACA so as to improve its convergence

properties (Vizine et al., 2005) and it is also adopted in our proposed approach (A^2CA). In a nutshell, a cooling schedule for the parameter that drives the picking probability k_p – Eq. (2) – is employed. The adopted scheme is simple: after one cycle (10,000 ant steps) has passed, the value of the parameter k_p starts being geometrically decreased, at each cycle, until it reaches a minimal allowed value, $k_{p\min}$, which corresponds to the stopping criterion for the algorithm. In the current implementation, k_p is cooled based on a geometric scheme presented in Eq. (4). It is important to emphasize that the SACA implementation used in this work also incorporates this *extra feature*, leading to the so-called SACA*. By doing so, more suitable and fair comparisons can be performed, in the sense that SACA* will also tend to converge to better clustering solutions.

$$\begin{aligned} k_p &\leftarrow k_p \times 0.98, \\ k_{p\min} &= 0.001. \end{aligned} \tag{4}$$

3.2 Progressive Vision

In SACA, the value of the density function, $f(i)$, given by Eq. (1), depends on the vision field, s^2, of each ant. The definition of a fixed value for s^2 may sometimes cause inappropriate behaviors, because a fixed perceptual area does not allow distinguishing between clusters of different sizes. A small area of vision implies a small perception of the cluster at a global level. Thus, small clusters and large clusters are all the same in this sense, for the agent only perceives a limited area of the environment. In some problems, the use of a too restrictive perception field may be limiting, whereas a too broad vision may cause undesirable merging of groups. On the one hand, even if a cluster is perfectly homogeneous (with identical elements) and sufficiently large, there still exists a small probability that an agent picks up a datum from the cluster and drops it somewhere else. On the other hand, a large vision field may be inefficient in the initial iterations, when the data elements are scattered at random on the grid, because analyzing a broad area may imply in analyzing a large number of small clusters simultaneously.

In order to overcome this difficulty, a progressive vision scheme was proposed for SACA as follows (Sherafat et al., 2004a). When an ant perceives a 'big' cluster, it increments its perception field (s_i^2) up to a maximal size. Now, s_i^2 is a specific parameter for each ant that will be dynamically and independently updated while running the algorithm. The question that remains is: 'How can an ant agent detect the size of a cluster so as to control the size of its vision field?'

We tackled this problem by using the density dependent function $f(i)$ as a control parameter. There is a relationship between the size of a cluster and its density dependent function: the average value of $f(i)$ increases as the clustering proceeds, and this happens because larger clusters tend to be formed. When $f(i)$ achieves a value greater than a pre-specified threshold θ, the parameter s^2 is incremented by n_s units until it reaches its maximum value.

$$\begin{aligned} &\text{If } f(i) > \theta \quad \text{and} \quad s^2 \le s^2{}_{\max}, \\ &\qquad \text{then } s^2 \leftarrow s^2 + n_s. \end{aligned} \tag{5}$$

where $s^2{}_{\max} = 7 \times 7$ and $\theta = 0.6$ in our implementation.

3.3 Pheromone Heuristics

In order to perform data clustering, the SACA takes into account the relative distance among all objects within the vision field of the ant. A problem with this approach is that it does not account for the work in progress at a global level. One form of overcoming this difficulty was proposed by Sherafat et al. (2004a,b). The method is based on the introduction of a local variable $\phi(i)$ associated with each bi-dimensional position, i, on the grid, such that the quantity of pheromone in that exact position becomes a function of the presence or absence of an object at i. Inspired by the way termites use pheromone to build their nests, the artificial agents in the modified ant clustering algorithm will add some pheromone to the objects they carry and this pheromone will be transferred to the grid when an object is deposited. During each iteration, the artificial pheromone $\phi(i)$ at each cell of the grid evaporates at a fixed rate.

Sherafat et al. (2004a,b) introduced a pheromone function, $Phe(\phi_{\max},\phi_{\min},P,\phi(i))$, given by Eq. (6), that influences the probability of picking up and dropping off objects from and on the grid. The proposed pheromone function varies linearly with the pheromone level at each grid position, $\phi(i)$, and depends on a number of user-defined parameters, such as the $\phi_{\max}$ and $\phi_{\min}$ values of pheromone perceived by the agent, and the maximal influence of pheromone allowed, P.

$$Phe(.) = \frac{2.P}{\phi_{max} - \phi_{min}}\phi(i) - \frac{2.P.\phi_{max}}{\phi_{max} - \phi_{min}} + P, \tag{6}$$

To accommodate the addition of pheromone on the grid, some variations on the picking and dropping probability functions of SACA were proposed in (Sherafat et al., 2004a,b), as described in Eqs. (7) and (8), respectively:

$$P_{pick}(i) = (1 - Phe(\phi_{\min}, \phi_{\max}, P, \phi(i))) \times \left(\frac{k_p}{k_p + f(i)}\right)^2. \tag{7}$$

$$P_{drop}(i) = (1 + Phe(\phi_{\min}, \phi_{\max}, P, \phi(i))) \times \left(\frac{f(i)}{k_d + f(i)}\right)^2. \tag{8}$$

where $\phi_{\max}$ represents the current largest amount of pheromone perceived by this agent; $\phi_{\min}$ corresponds to the current smallest amount of pheromone perceived by this agent; P is the maximum influence of the pheromone in changing the probability of picking and dropping data elements; and $\phi(i)$ is the quantity of pheromone in the current position i.

Note that in Eq. (8) the dropping probability originally derived from the model of Deneubourg et al. (1991) was employed. Basically, this choice was made because the algorithm presented superior performance when using the function proposed by Deneubourg et al. (1991) – given by Eq. (9) - instead of Eq. (3) for the dropping probability. This was also the case for SACA. Therefore,

we also adopt this strategy in our present work, namely the dropping probability is an inverse function of a parameter k_d:

$$P_{drop}(i) = \left(\frac{f(i)}{k_d + f(i)} \right)^2 . \quad (9)$$

Based on the sensitivity analysis described in Sherafat et al. (2004a,b) and on some preliminary experiments, we realized that setting the parameters $\phi_{\max}$, $\phi_{\min}$ and P may become a difficult task depending on the problem at hand. In order to reduce the number of user-defined parameters and to improve even further the performance of the algorithm, we propose to substitute Eqs. (7) and (8) by the following equations:

$$P_{pick}(i) = \frac{1}{f(i)\phi(i)} \left(\frac{k_p}{k_p + f(i)} \right)^2 . \quad (10)$$

$$P_{drop}(i) = f(i)\phi(i) \left(\frac{f(i)}{k_d + f(i)} \right)^2 . \quad (11)$$

where $f(i)$ is the density dependent function, $\phi(i)$ is the quantity of pheromone in the current position i, and k_p and k_d are the picking and dropping probability constants, respectively. Note that, in this new proposal, the only new parameter introduced in relation to SACA is the pheromone level at each position of the grid.

According to Eq. (10), the probability that an ant picks up an item from the grid is inversely proportional to the amount of pheromone at that position and also to the density of objects around i. This equation thus accounts for the pheromone reinforcement signal in regions of the space filled with similar objects. If the region is filled with dissimilar objects, however, the incorporation of $f(i)$ multiplying $\phi(i)$ counterbalances the effects of eventual high pheromone concentrations. By the same token, Eq. (11) states that regions with high concentration levels of pheromone are attractive for the deposition of more objects of similar type.

It is important to observe that a region with a high quantity of pheromone tends to be either a recently constructed cluster or a cluster under construction. The pheromone is a variable of the discrete grid environment, i.e. each grid position i has an independent variable $\phi(i)$ for which pheromone evaporation and diffusion procedures are implemented. The rate at which pheromone evaporates is preset, as defined in Eq. (12). Each grid position i also has a connection to its neighbors that causes a percentage of $\phi(i)$ to be diffused to them. This is performed in such a way that the pheromone percentage for the two closer neighbors in all directions decays geometrically in the reason of 1/2, whereas for the third closer neighbors in all directions it is set equal to zero. In our implementation, the maximum amount of added pheromone $\phi(i)$ is equal to 0.01. The proposed approach increases the probability of deconstruction of relatively small clusters and increases the probability of dropping data elements in denser clusters. This is directly influenced by the similarity between the data and the cluster. This proposal then becomes a sort of density-based clustering procedure (Everitt et al., 2001).

$$\phi(i) \leftarrow \phi(i) \times 0.99. \quad (12)$$

4 Performance Evaluation

In order to assess the performance of the adaptive ant-clustering algorithm (A^2CA) in comparison with the standard algorithm with cooling and dropping probability given by Eq. (9), named here SACA*, both algorithms were applied to a number of synthetic data sets and to one real-world bioinformatics data set. The parameters used to run the algorithms were based on the sensitivity analysis performed in Sherafat et al. (2004a) and on some preliminary experiments performed here. The benchmarks used for evaluation and the respective adaptation parameters for the algorithms are summarized below. Further details are provided in each dedicated section. Parameters $\theta = 0.6$, $k_p = 0.20$, $k_d = 0.05$ are assumed default and were chosen for all experiments.

- 4Gauss: 100 objects divided into 4 clusters (classes). $n_{ants} = 10$, grid = 25×25, and $\alpha = 0.35$.
- Ruspini data: 75 objects divided into 4 classes. $n_{ants} = 10$, grid = 25×25, and $\alpha = 0.35$.
- ANIMALS data set: 16 objects with 13 attributes (the number of classes varies based on the grouping performed). $n_{ants} = 1$, grid = 15×15, and $\alpha = 2.10$.
- Yeast galactose data: 205 objects divided into 4 classes. $n_{ants} = 10$, grid = 35×35, and $\alpha = 1.05$.

Note that the parameters used to run the algorithms are almost the same for all data sets; the only ones that change are α, the grid size, and the number of ants n_{ants}. As one grid cell is used to accommodate one object, the grid is increased in size in proportion to the size of the input data set. The parameter α, by contrast, weighs the influence of the distance measure in determining the clusters. Its value was linearly varied using a factor 0.35 for the employed data sets. In the ANIMALS data set, a single ant was used because the number of objects is very small, only 16.

4.1 Four Gaussian Distributions

The first data set used to illustrate the performance of the algorithm was a modified version of the well-known four classes data set proposed by Lumer and Faieta (1994) to study the standard ant-clustering algorithm. The data set used here corresponds to four distributions of 25 data points each, defined by Gaussian probability density functions with various means μ and fixed standard deviation $\sigma = 1.5$, $G(\mu,\sigma)$, as follows (Figure 1):

$A = [x \propto G(0,1.5), y \propto G(0,1.5)]$;
$B = [x \propto G(0,1.5), y \propto G(8,1.5)]$;
$C = [x \propto G(8,1.5), y \propto G(0,1.5)]$;
$D = [x \propto G(8,1.5), y \propto G(8,1.5)]$.

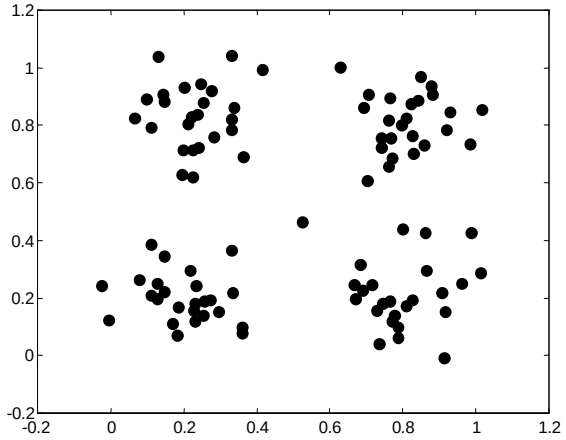
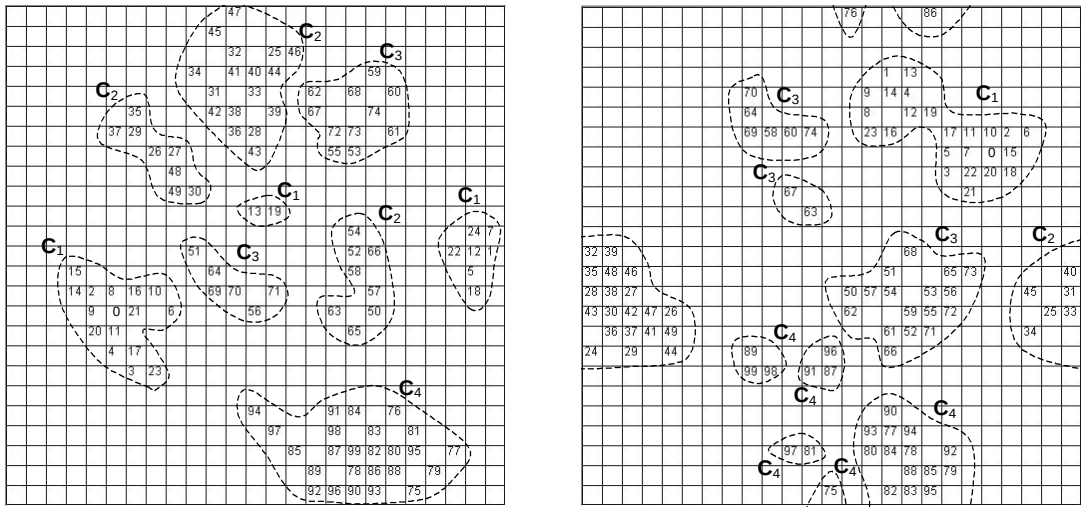


Figure 1: Gaussian distributions: input data set.

schedule for k_p described previously (SACA*). The pictures correspond to the output grid of two different simulations generated by the ants after convergence, in this case after 273,000 ant steps (27.3 cycles). Each input datum is numbered from 0 to 99, where the first 25 (from 0 to 24) belong to the first cluster, and so on. Note that, accordingly with what was previously discussed by Lumer and Faieta (1994), the standard ant-clustering algorithm (SACA), though capable of correctly clustering the data, generates a large number of sub-clusters in most cases. In our experiments, we observed that, even with the use of a cooling procedure (i.e., SACA*), this characteristic tends to be maintained. Figure 2(b) shows some results for A^{2}CA. It can be noted that the adaptive algorithm generates a much smaller number of sub-clusters; in most cases, only four or five groups of data are generated.

Figure 2(a) depicts some simulation results for the standard ant-clustering algorithm with the geometric cooling



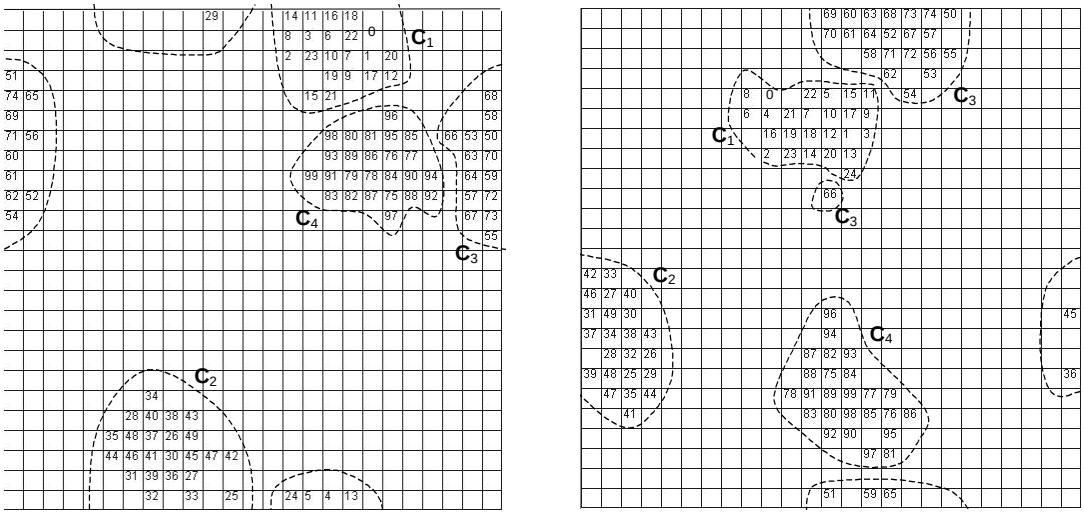
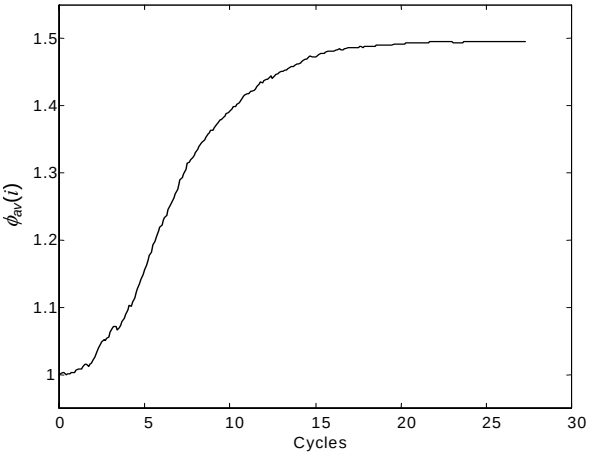
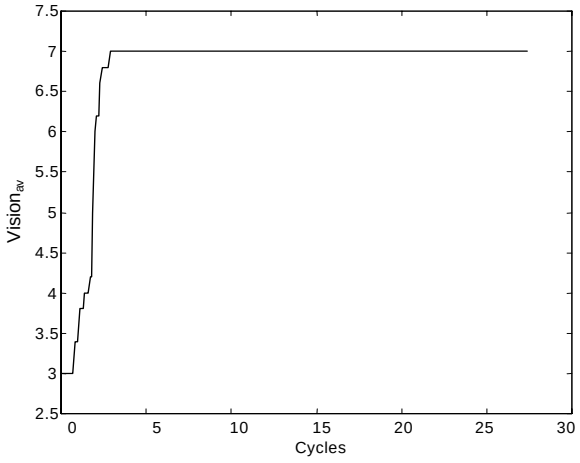


Figure 2: Two different results for the standard ant-clustering algorithm SACA* (a) and A^{2}CA (b).

Figure 3(a) and (b) show, respectively, the evolution of the average pheromone level on the grid and the average vision of all ants for the simulations depicted in Figure 2(b-1). In Figure 4(a) we reproduce Figure 2(b-1), for convenience, and contrast the final distribution of objects onto the grid with the 3D (Figure 4(b)) and 2D (Figure 4(c)) views of the pheromone distribution on the grid after convergence. It is easy to observe the higher concentration of pheromone in regions of the grid with large data density. It can also be noted from these pictures that the average pheromone level on the grid and vision field of the ants tend to stabilize after a number of iterations. In the particular case of vision, all ants converge to a vision field of dimension 7×7.

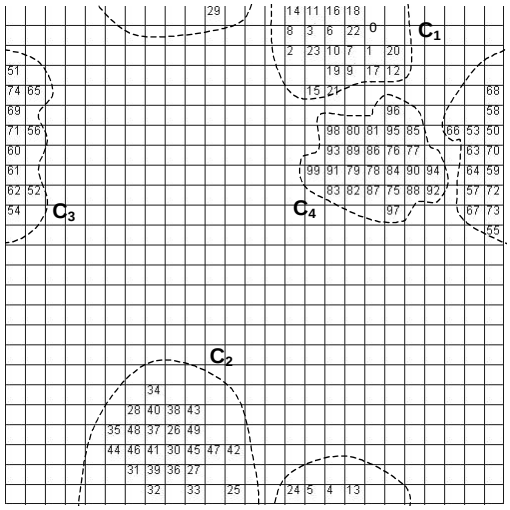


(a)

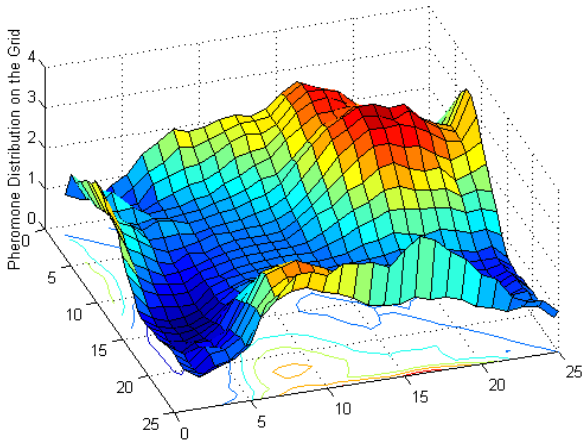


(b)

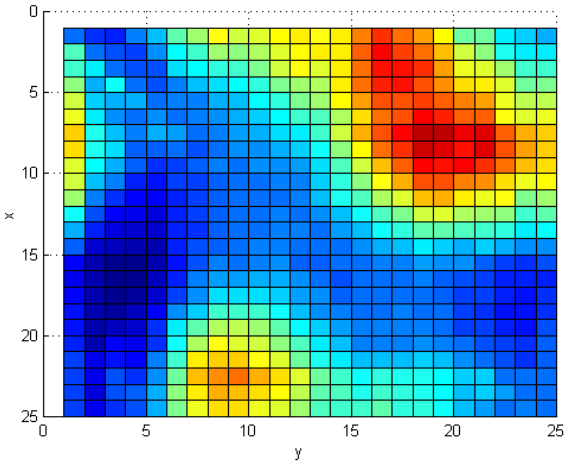
Figure 3: Evolution of the average pheromone level on the grid (a), and the average vision field of the ants (b) for the experiment depicted in Figure 2(b-1).



(a)



(b)



(c)

Figure 4: Objects and pheromone distribution on the grid after convergence. (a) Final distribution of objects on the grid after convergence (Figure 2(b-1)). Three-dimensional perspective (b) and two-dimensional perspective (c) of the pheromone distribution on the grid after convergence.

4.2 Animals Data Set

This section compares the performance of A^{2}CA with SACA* when applied to the ANIMALS data set. This high-dimensional data set was originally proposed by Ritter and Kohonen (1989) to verify the capability of a *self-organizing map* creating a topographic map of the input data based on a symbol set. The data set is composed of 16 input vectors, each representing an animal with the binary feature attributes as shown in Table 1. A value of 1 in this table corresponds to the presence of an attribute, whilst a value of 0 corresponds to the lack of this attribute. The authors suggested that the interestingness of this data set lies in the fact that the relationship between the different symbols may not be directly detectable from their encoding, thus not presuming any metric relations even when the symbols represent similar items.

Table 1: Animal data set with their names and binary attributes (after Ritter & Kohonen, 1989).

		0. Dove	1. Hen	2. Duck	3. Goose	4. Owl	5. Hawk	6. Eagle	7. Fox	8. Dog	9. Wolf	10. Cat	11. Tiger	12. Lion	13. Horse	14. Zebra	15. Cow
Is	Small	1	1	1	1	1	1	0	0	0	0	1	0	0	0	0	0
	Medium	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0
	Big	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1
Has	Two legs	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
	Four legs	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
	Hair	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
	Hooves	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
	Mane	0	0	0	0	0	0	0	0	0	1	0	0	1	1	1	0
	Feathers	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
Likes to	Hunt	0	0	0	0	1	1	1	1	0	1	1	1	1	0	0	0
	Run	0	0	0	0	0	0	0	0	1	1	0	1	1	1	1	0
	Fly	1	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0
	Swim	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0

Table 2 describes the results found by both algorithms when applied to the ANIMALS data set. It can be observed that A^{2}CA consistently determined two groups of data, one corresponding to the birds and another referring to the mammals. In most cases SACA* presented the same results as A^{2}CA, but it sometimes separated the mammals into two groups that apparently do not make much sense. For instance, in run 5, SACA* mixed Lion (12) with Horse (13) and Zebra (14). In (Haykin, 1999 – p. 476), a self-organizing map for the ANIMALS data set is presented with three main groups: birds, peaceful mammals and hunters. However, the partition of the output map could also have been made so as to distinguish only two different groups, as the results presented by SACA* and A^{2}CA.

Table 2: Groups found by SACA* and A^{2}CA for the ANIMALS data set.

	SACA*		**A^{2}CA**	
Run	N_c	**Groups**	N_c	**Groups**
1	2	(0-6) (7-15)	2	(0-6) (7-15)
2	2	(0-6) (7-15)	2	(0-6) (7-15)
3	2	(0-6) (7-15)	2	(0-6) (7-15)
4	3	(0-6) (10) (7-9,11-15)	2	(0-6) (7-15)
5	3	(0,6) (7-11,15) (12-14)	2	(0-6) (7-15)
6	2	(0-6) (7-15)	2	(0-6) (7-15)
7	3	(0-6) (7-12,15) (13,14)	2	(0-6) (7-15)
8	2	(0-6) (7-15)	2	(0-6) (7-15)
9	2	(0-6) (7-15)	2	(0-6) (7-15)
10	2	(0-6) (7-15)	2	(0-6) (7-15)
Av. ± std	2.3 ± 0.48		2 ± 0	

4.3 Ruspini Data

The Ruspini data is a well-known dataset commonly used to benchmark clustering algorithms (Kaufman & Rousseeuw, 1990). It is formed by 75 objects grouped into four clusters, as depicted in Figure 5. Let n_c be the number of clusters found and P_{mc} the percentage of misclassification. Table 3 summarizes the performance of both algorithms when applied to the Ruspini data. The

results presented are the average ± standard deviation taken during 10 runs of each algorithm. Similarly to the results presented in the previous experiments, A^{2}CA consistently found the correct number of clusters with no classification errors.

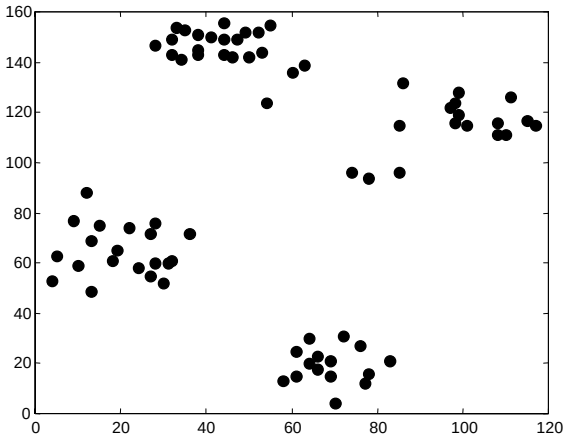


Figure 5: Ruspini data.

Table 3: Performance evaluation for the standard ant clustering algorithm with cooling (SACA*) and the adaptive ant clustering algorithm (A^{2}CA).

	SACA*		**A^{2}CA**	
	n_c	P_{mc} (%)	n_c	P_{mc} (%)
Ruspini	7.4 ± 1.46	1.5 ± 2.72	4.0 ± 0.0	0 ± 0.0

4.4 Yeast Galactose Data

The last data used for evaluation is the *yeast galactose* data set (Yeung et al., 2003). This is a real-world bioinformatics dataset composed of 20 experiments (attributes) – nine single-gene deletions and one wild-type experiment with galactose and raffinose, nine deletions and one wild-type without galactose and raffinose. Similarly to Yeung et al. (2003), we used a subset of 205 genes (objects), whose expression patterns reflect four functional categories (clusters) formed by 83, 15, 93 and 14 genes (objects). The dataset used in the simulations reported here take into account four repeated measurements, what may yield more accurate and more stable clusters (Yeung et al., 2003). To cluster data with repeated measurements, the average expression levels over all repeated measurements for each gene and each experiment were taken.

For this data set, the standard algorithm (SACA*) demonstrated to be incapable of correctly grouping the data in most simulations. The proposed algorithm, however, was capable of appropriately grouping the data in all runs, but with varying numbers of clusters being found each time the algorithm was run. Over 10 runs, A^{2}CA presented the following results: $n_c = 6.9 \pm 1.0$ and $P_{mc} = 3.17\% \pm 0.93\%$. Figure 6 depicts one solution for the A^{2}CA applied to the yeast data set. This figure also depicts the clusters found (within dashed lines) and the objects incorrectly grouped (within solid lines).

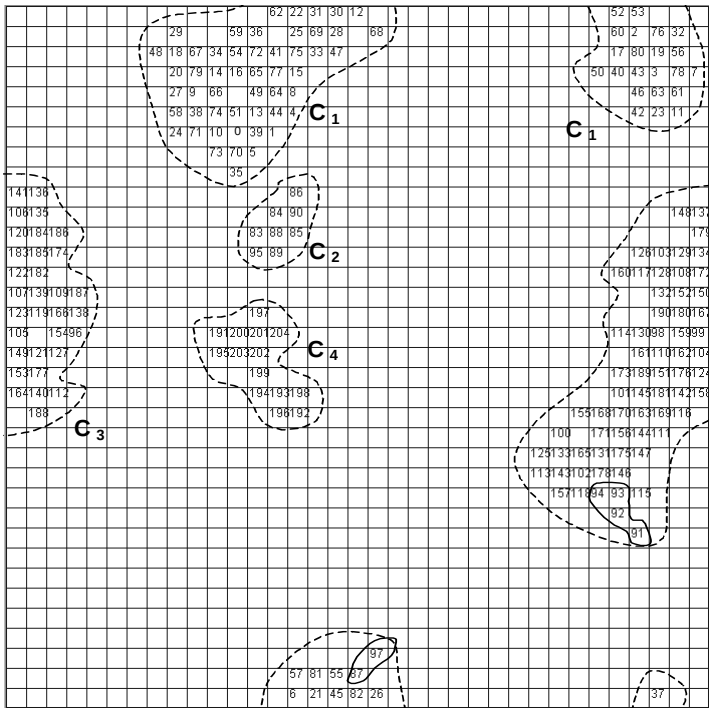


Figure 6: One grid solution for A^{2}CA when applied to the yeast galactose data.

5 Ant Clustering Algorithms: A Brief Survey

Several clustering methods based on ant behavior have been proposed in the literature, showing the increasing importance of this subject. This section provides a brief description of these methods, following a chronological order.

In 1991, Deneubourg et al. (1991) introduced a model in which simple ants were able to sort into piles objects initially strewn randomly across a plane. These ants have a sorting behavior based on local rules, i.e. possessing only local perceptual capabilities. Gutowitz (1993) called these agents *basic ants*, which have: (i) a finite memory, which is a register of length n that records the presence or absence of objects at the ant's previous n locations; (ii) an object-manipulation capacity; (iii) a function that gives the probability to manipulate an object proportionally to the values in memory and a random variable; and (iv) the capability to execute Brownian motion. Besides, as previously observed in the Deneubourg's model, two objects can only be either identical or different. Obviously, this same idea can be easily extended to deal with other distance metrics such as the well-known Euclidean norm.

Although the basic ants have only local perceptual capabilities, they are able to promote global order. The mechanism underlying this phenomenon was carefully investigated by Gutowitz (1993). He proposed the *complexity-seeking ants*, which are variants of the *basic ants* proposed by Deneubourg et al. (1991). The complexity-seeking ants are allowed to *see* local complexity and tend to perform actions in regions of highest local complexity. The neighborhood complexity is the number of faces that separate cells of different types, containing or not an object. In this sense, all-empty or all-occupied neighborhoods have *zero* complexity (low entropy), whereas checkerboard patterns have complexity equals to 12 (assuming a 9-cell neighborhood). Thus, complexity-seeking ants can calculate the complexity of their local environment and are able to accomplish their task more efficiently than the *basic ants*, mainly because they tend to manipulate objects in regions of high complexity; that is, at intermediate density regions, where the entropy is high.

As previously addressed in Section 2, Lumer and Faieta (1994) introduced a method for structuring complex datasets into clusters. The proposed method is inspired by the model of Deneubourg et al. (1991), in which ant-like agents move at random on a 2-dimensional grid, where objects are scattered at random. Inspired by the biological phenomenon of dead body clustering, the ants do not communicate with each other and can only perceive their surrounding local environment. In this context, each ant-like agent can either pick up an object from the grid or drop it onto the grid. The probability of picking up an object decreases with both the density of other objects and the similarity with other objects within a given neighborhood. By contrast, the probability of dropping an object increases with the similarity and the density of objects within a local region. Although the work in (Deneubourg et al., 1991) is restricted to environments made of either identical objects or two distinct types of objects, Lumer and Faieta (1994) generalized this model to work with objects that differ along a continuous similarity measure. This led to the algorithm that we have called SACA in our work.

Monmarché et al. (1999) combined the stochastic and exploratory principles of clustering ants with the deterministic and heuristic principles of the popular k-means algorithm in order to improve the convergence of the ant-based clustering algorithm. The proposed hybrid method is called AntClass and is based on the work of Lumer and Faieta (1994). The AntClass algorithm allows an ant to drop more than one object in the same cell, forming heaps of objects. It involves four main steps: (i) ant-based clustering; (ii) k-means algorithm using the initial partition provided by ants; (iii) ant-based clustering on heaps of objects previously found; (iv) k-means algorithm once more. Another important contribution of the AntClass algorithm is that it also makes use of hierarchical clustering, implemented by allowing ants to carry an entire heap of objects.

Ramos and Merelo (2002) developed an ant clustering system called ACLUSTER, which was employed for textual document clustering. The authors proposed the use of bio-inspired spatial transition probabilities, avoiding randomly moving agents, which may explore non-interesting regions. In this sense, ants do not move randomly like in SACA, but according to transition probabilities that depend on the spatial distribution of pheromone across the environment. If a particular cluster disappears, the pheromone tends to evaporate from that location. This approach is interesting, because pheromone represents the swarm memory and all ants can benefit from it. In other words, the ants share a common memory. Another important difference in relation to the SACA refers to the use of combinations of two independent response threshold functions; each associated with different environmental factors, namely, the number of objects in the neighborhood and their similarity. The ACLUSTER algorithm was also employed into a digital image retrieval problem, and further details about a case study within a granite database can be found in (Ramos et al., 2002). In a later work, Abraham and Ramos (2003) applied the ACLUSTER to discover Web usage patterns and thereafter a genetic programming approach to analyze the visitor trends.

Handl and Meyer (2002) employed ant-based clustering as the core of a visual document retrieval system for worldwide web searches in which the basic goal is to classify online documents by contents' similarity. The authors adopted an idea of short-term memory and employed ants with different speeds, also allowing them to *jump*. In addition, they introduced an adaptive scaling strategy, as well as some further modifications to achieve reliable results and to improve efficiency. The proposed method starts with a very fine distinction between data elements and reduces it only if necessary; that is, if after a pre-defined number of steps only few dropping or picking up occur. The authors also adopted a stagnation con-

trol similar to the one described in Monmarché et al. (1999), in which after a pre-defined number of unsuccessful dropping attempts an ant drops its load regardless of the neighborhood's similarity. Finally, Handl and Meyer (2002) used *eager ants*, which take objects immediately after dropping their loads.

Labroche et al. (2002) proposed a clustering algorithm, called ANTCLUST, based on a modeling of the chemical recognition system of ants. This system allows the construction of a colonial odor used for determining the ants' nest membership, such that ants can discriminate between nest mates and intruders. In the ANTCLUST, each object is assigned to an artificial ant and represents part of the ant's odor. At the beginning of the clustering process, ants are under the influence of any nest and consequently have no *label* (representative of the nest). Then, random meetings between ants are simulated and *labels* are updated according to behavioral rules, which take into account the similarity among data. These *labels* evolve over time until each ant has found its best nest, providing a partition of the objects.

Kanade and Hall (2003) combined the ant based clustering algorithm proposed by Monmarché et al. (1999) with the classical *Fuzzy C-Means* algorithm (FCM) (Bezdek, 1981). The ant based clustering algorithm is employed to initially create raw clusters, which are then refined by the FCM algorithm. In this sense, the corresponding centroids of each initial cluster are taken as initial prototypes for the FCM. Then, each object is assigned to its best matching fuzzy cluster, i.e. the cluster it has the highest membership to. These new clusters can be moved and merged by the ants. Finally, the obtained clusters are also refined by the FCM.

Handl et al. (2003) proposed a scheme that enables an unbiased interpretation of the clustering solutions obtained by ant based clustering algorithms. The authors argue that although many of the results obtained by ant algorithms *look* promising, there is a lack of knowledge about the actual performance of such algorithms, i.e. in general, the evaluation of the results has been performed by means of visual observation. In order to overcome this limitation, they propose a technique that allows converting the *implicit* clusters found by an ant algorithm into an *explicit* data partitioning. The proposed technique is based on the application of an agglomerative hierarchical clustering method to the positions of the data items on the grid. Taking into consideration the developed method, the results achieved by the ant-based clustering algorithm proposed by Handl and Meyer (2002) are compared, using both synthetic and real datasets, with those obtained by two classical algorithms (*k*-means and agglomerative average link), showing that the ant-based algorithm performs well when compared with them.

6 Conclusions and Future Work

The ant-clustering algorithm is a self-organizing multi-agent system typically used for clustering unlabelled datasets. Its goal is to project the original data into a bi-dimensional output grid and position those items that are similar to each other in their original space of attributes in neighbor regions of the output grid. By doing this, the algorithm is capable of grouping together items that are similar to each other and presenting the result of this grouping process on a bi-dimensional display (2D grid) that can be easily inspected visually helping the user to deal with the overload of information. The advantage of visual data exploration is that the user is directly involved in the data mining process (Keim, 2002). This results in a device suitable for exploratory data analysis even when the input data set lies in a high-dimensional space.

This paper provided a number of contributions to the field in two main frontlines. First, several modifications were introduced in the standard ant-clustering algorithm so as to enhance its performance and convergence properties. In particular, we proposed a cooling schedule for the parameter that controls the rate of picking up objects from the grid. This guarantees that the algorithm always stabilizes after a number of iteration steps. Furthermore, we developed the ideas of progressive vision (Sherafat et al., 2004a) and proposed a new form of implementing the pheromone heuristics on the grid in such a way that groups of data reinforce the attraction to those regions of the grid that contain data. The second contribution of this article was the presentation of a review from the literature citing and briefly describing most works and applications of ant clustering algorithms to date. The proposed adaptive algorithm, named A^2CA, was applied to a number of benchmark data sets and to a real world bioinformatics data set. The obtained results were compared to the standard ant clustering algorithm with cooling schedule and modified dropping probability, and stress the benefits of the modifications introduced in the proposed algorithm. Most importantly, A^2CA demonstrated a good robustness in terms of finding the correct number of clusters in the data set, low variations of the results in terms of number of clusters found, and always stabilized after a fixed number of iterations automatically defined by the algorithm.

Despite the encouraging results presented here, there are still several avenues for investigation that deserve to be pursued. For instance, an automatic form of segmenting the output grid and counting the number of clusters found after convergence can be proposed; the algorithm can be transformed into a supervised algorithm, that is, information about a set of known classes of data can be used to aid the definition of the final configuration of the grid; a hierarchical analysis of the input data can be proposed by systematically varying some of the user-defined parameters; the use of heaps of objects instead of a one-object-one-grid-position scheme used here can be performed (though we believe that the addition of pheromone to the grid may compensate for the effect of allowing heaps of objects to be formed); the use of local search procedures (e.g., k-means) to fine tune the clusters found by the ants; and a sensitivity analysis in relation to the user-defined parameters can be performed.

Acknowledgement

The authors thank UniSantos, CNPq and FAPESP for the financial support.

References

[1] Abraham, A., Ramos, V. (2003). Web Usage Mining Using Artificial Ant Colony Clustering and Genetic Programming. Proc. of the Congress on Evolutionary Computation (CEC 2003), Canberra, pp. 1384-1391, IEEE Press.

[2] Bezdek, J.C., (1981). Pattern Recognition with Fuzzy Objective Function Algorithm, Plenum Press.

[3] Bonabeau, E., Dorigo, M. and Théraulaz, G. (1999). Swarm Intelligence from Natural to Artificial Systems: Oxford University Press.

[4] Camazine, S., Deneubourg, J.-L., Franks, N. R., Sneyd, J., Theraulaz, G. and Bonabeau, E. (2001). Self-Organization in Biological Systems: Princeton University Press.

[5] de Castro, L. N. & Von Zuben, F. J. (2004), *Recent Developments in Biologically Inspired Computing*, Idea Group Inc.

[6] Deneubourg, J. -L., Goss, S., Sendova-Franks, N., A., Detrain, C. and Chrétien, L. (1991). The Dynamics of Collective Sorting: Robot-Like Ant and Ant-Like Robot. In J. A. Meyer and S. W. Wilson (eds.). Simulation of Adaptive Behavior: From Animals to Animats: MIT Press/Bradford Books, 356-365.

[7] Everitt, B.S., Landau, S., Leese, M., (2001). Cluster Analysis: Arnold Publishers, London.

[8] Gutowitz, H. (1993). Complexity-Seeking Ants. Proceedings of the Third European Conference on Artificial Life.

[9] Handl, J., Knowles, J., Dorigo, M. (2003). On the performance of ant-based clustering. Proc. of the 3rd International Conference on Hybrid Intelligent Systems, Design and Application of Hybrid Intelligent Systems, pp. 204-213, IOS Press.

[10] Handl, J., Meyer, B. (2002). Improved Ant-Based Clustering and Sorting in a Document Retrieval Interface. In J.J. Merelo, J.L.F. Villacañas, H.G. Beyer, P. Adamis Eds.: Proceedings of the PPSN VII – 7th Int. Conf. on Parallel Problem Solving from Nature, Granada, Spain, Lecture Notes in Computer Science 2439, pp. 913-923, Springer-Verlag, Berlin.

[11] Kanade, P., Hall, L.O. (2003). Fuzzy ants as a clustering concept. Proc. of the 22nd International Conference of the North American Fuzzy Information Processing Society (NAFIPS), pp. 227-232.

[12] Kaufman, L., Rousseeuw, P.J. (1990), Finding Groups in Data – An Introduction to Cluster Analysis, Wiley Series in Probability and Mathematical Statistics, John Wiley & Sons Inc.

[13] Keim, D.A. (2002), Information Visualization and Visual Data Mining: IEEE Transactions on Visualization and Computer Graphics, vol. 7, n.1, pp. 100-107.

[14] Kennedy, J., Eberhart, R. and Shi. Y. (2001). Swarm Intelligence: Morgan Kaufmann Publishers.

[15] Labroche, N., Monmarché, N., Venturini, G. (2002). A new clustering algorithm based on the chemical recognition system of ants. Proc. of the 15th European Conference on Artificial Intelligence, France, pp. 345-349, IOS Press.

[16] Lumer, E.D. and Faieta, B. (1994). Diversity and Adaptation in Populations of Clustering Ants. Proceedings of the Third International Conference On the Simulation of Adaptive Behavior: From Animals to Animats 3: MIT Press, 499-508.

[17] Monmarché, N., Slimane, M., Venturini, G., (1999). On Improving Clustering in Numerical Databases with Artificial Ants. Advances in Artificial Life, D. Floreano, J.D. Nicoud, and F. Mondala Eds., Lecture Notes in Computer Science 1674, pp. 626-635, Springer-Verlag, Berlin.

[18] Paton, R. (Ed.) (1994). Computing with Biological Metaphors: Chapman & Hall.

[19] Ramos, V., Merelo, J.J.. (2002). Self-Organized Stigmergic Document Maps: Environment as a Mechanism for Context Learning. In E. Alba, F. Herrera, J.J. Merelo et al. Eds., AEB´2002, First Spanish Conference on Evolutionary and Bio-Inspired Algorithms, 284-293, Spain.

[20] Ramos, V., Muge, F., Pina, P. (2002). Self-Organized Data and Image Retrieval as a Consequence of Inter-Dynamic Synergistic Relationships in Artificial Ant Colonies. In J. Ruiz-del-Solar, A. Abrahan and M. Köppen Eds., Soft-Computing Systems - Design, Management and Applications, Frontiers in Artificial Intelligence and Applications: IOS Press, v. 87, 500-509, Amsterdam.

[21] Ritter, H. & Kohonen, T. (1989). Self-Organizing Semantic Maps. *Biol. Cybern.*, **61**, pp. 241-254.

[22] Sherafat, V., de Castro, L. N. & Hruschka, E. R. (2004a). TermitAnt: An Ant Clustering Algorithm Improved by Ideas from Termite Colonies. In Proc. of ICONIP 2004, Special Session on Ant Colony and Multi-Agent Systems, Lecture Notes in Computer Science, v. 3316, pp. 1088-1093.

[23] Sherafat, V., de Castro, L. N. & Hruschka, E. R. (2004b). The Influence of Pheromone and Adaptive Vision on the Standard Ant Clustering Algorithm. In: L. N. de Castro and F. J. Von Zuben, *Recent Developments in Biologically Inspired Computing*, Chapter IX, pp. 207-234. Idea Group Inc.

[24] Vizine, A. L., de Castro, L. N., Gudwin, R. R. (2005). Text Document Classification using Swarm Intelligence. In Proc. of KIMAS 2005, CD ROM.

[25] Yeung, K.Y., Medvedovic, M., Bumgarner, R.E. (2003), Clustering gene-expression data with repeated measurements, Genome Biology, v.4, issue 5, article R34.

Efficient Pre-Processing for Large Window-Based Modular Exponentiation Using Ant Colony

Nadia Nedjah
Department of Electronics Engineering and Telecommunications,
Faculty of Engineering, State University of Rio de Janeiro, Brazil
nadia@eng.uerj.br

Luiza de Macedo Mourelle
Department of Systems Engineering and Computation,
Faculty of Engineering, State University of Rio de Janeiro, Brazil
ldmm@eng.uerj.br

Keywords: ant colony, addition chain, cryptosystem, modular exponentiation

Received: September 30, 2004

Modular exponentiation is the main operation to RSA-based public-key cryptosystems. It is performed using successive modular multiplications. This operation is time consuming for large operands, which is always the case in cryptography. For software or hardware fast cryptosystems, one needs thus reducing the total number of modular multiplications required. Existing methods attempt to reduce this number by partitioning the exponent in constant or variable size windows. However, these window-based methods require some pre-computations, which themselves consist of modular exponentiations. It is clear that pre-processing needs to be performed efficiently also. In this paper, we exploit the ant colony strategy to finding an optimal addition sequence that allows one to perform the pre-computations in window-based methods with a minimal number of modular multiplications. Hence we improve the efficiency of modular exponentiation. We compare the yielded addition sequences with those obtained using Brun's algorithm.

Povzetek: Metoda kolonij mravelj je uporabljena za kriptografske probleme.

1 Introduction

Public-key cryptographic systems (such as the RSA encryption scheme [6], [12]) often involve raising large elements of some groups fields (such as GF(2^n) or elliptic curves [9]) to large powers. The performance and practicality of such cryptosystems is primarily determined by the implementation efficiency of the modular exponentiation. As the operands (the plain text of a message or the cipher (possibly a partially ciphered) are usually large (i.e. 1024 bits or more), and in order to improve time requirements of the encryption/decryption operations, it is essential to attempt to minimise the number of modular multiplications performed.

A simple procedure to compute $C = T^E \bmod M$ based on the paper-and-pencil method is described in Algorithm 1. This method requires E-1 modular multiplications. It computes all powers of T: $T \rightarrow T^2 \rightarrow \ldots \rightarrow T^{E-1} \rightarrow T^E$.

Algorithm 1. simpleExponentiation(T, M, E)
1. $C := T$;
2. for $i := 1$ to $E - 1$ do $C := (C \times T) \bmod M$;
3. return C;
end algorithm.

The computation of exponentiations using Algorithm 1 is very inefficient. The problem of yielding the power of a number using a minimal number of multiplications is NP-hard [5], [10]. There are several efficient algorithms that perform exponentiation with a nearly minimal number of modular multiplications, such that the window-based methods. However, these methods need some pre-computations that if not performed efficiently can deteriorate the algorithm overall performance. The pre-computations are themselves an ensemble of exponentiations and so it is also NP-hard to perform them optimally. In this paper, we concentrate on this problem and engineer a new way to do the necessary pre-computations very efficiently. We do so using the ant colony methodology. We compare our results with those obtained using the Brun's algorithm [1].

Ant systems [2-1] are distributed multi-agent systems [3-1] that simulate real ant colony. Each agent behaves as an ant within its colony. Despite the fact that ants have very bad vision, they always are capable to find the shortest path from their nest to wherever the food is. To do so, ants deposit a trail of a chemical substance called *pheromone* on the path they use to reach the food. On intersection points, ants tend to choose a path with high amount of pheromone. Clearly, the ants that travel through the shorter path are ca-

pable to return quicker and so the pheromone deposited on that path increases relatively faster than that deposited on much longer alternative paths. Consequently, all the ants of the colony end using the shorter way.

In this paper, we exploit the ant colony methodology to obtain an optimal solution to AS-chain minimisation *NP*-complete problem. In order to clearly report the research work performed, we subdivide the rest of this paper into five important sections. In Section 2, we present the window methods; In Section 3, we present the concepts of addition chains and sequence and they can be used to improve the pre-computations of the window methods; In Section 4, we give an overview on ant colony concepts; In Section 5, we explain how these concepts can be used to compute a minimal addition chain to perform efficiently necessary pre-computations in the window methods. In Section 6, we present some useful results.

2 Window-Based Methods

Generally speaking, the window methods for exponentiation [5] may be thought of as a three major step procedure:

1. partitioning in k-bits windows the binary representation of the exponent E;
2. pre-computing the powers in each window one by one;
3. iterating the squaring of the partial result k times to shift it over, and then multiplying it by the power in the next window when if window is not 0.

There are several partitioning strategies. The window size may be constant or variable. For the m-ary methods, the window size is constant and the windows are next to each other. On the other hand, for the sliding window methods the window size may be of variable length. It is clear that zero-windows, i.e. those that contain only zeros, do not introduce any extra computation. So a good strategy for the sliding window methods is one that attempts to maximise the number of zero-windows. The details of m-ary methods are exposed in Section 2.1 while those related to sliding constant-size window methods are given in Section 2.2. In Section 2.3, we introduce the adaptive variable-size window methods.

2.1 M-ary Methods

The m-ary methods [3] scans the digits of E form the less significant to the most significant digit and groups them into partitions of equal length $\log_2 m$, where m is a power of two. Note that 1-ary methods coincides with the square-and- multiply well-known binary exponentiation method.

In general, the exponent E is partitioned into p partitions, each one containing $l = \log_2 m$ successive digits. The ordered set of the partition of E will be denoted by $\mathbb{P}(E)$. If the last partition has less digits than $\log_2 m$, then the exponent is expanded to the left with at most $log_2 m - 1$ zeros. The m-ary algorithm is described in Algorithm 2, wherein V_i denotes the decimal value of partition P_i.

Algorithm 2. m-aryMethod(T, M, E)
1. Partition E into p l-digits partitions;
2. for $i := 2$ to m Compute $T^i \bmod M$;
3. $C := T^{V_p} \bmod M$;
4. for $i := p - 2$ downto 0
5. $\quad C := C^{2^l} \bmod M$;
6. $\quad$ if $V_i \neq 0$ then $C := C\times \bmod M$;
7. return C;

end algorithm.

2.2 Sliding Window Methods

For the sliding window methods the window size may be of variable length and hence the partitioning may be performed so that the number of zero-windows is as large as possible, thus reducing the number of modular multiplication necessary in the squaring and multiplication phases. Furthermore, as all possible partitions have to start (i.e. in the right side) with digit 1, the pre-processing step needs to be performed for odd values only. The sliding method algorithm is presented in Algorithm 3, wherein d denotes the number of digits in the largest possible partition and L_i the length of partition P_i.

Algorithm 3. slidingWindowMethod(T, M, E)
1. Partition E using the given strategy;
2. for $i := 2$ to $2^d - 1$ step 2 do
3. $\quad$ Compute $T^i \bmod M$;
4. $C := T^{V_{p-1}} \bmod M$;
5. for $i := p - 2$ downto 0 do
6. $\quad C := C^{L_i} \bmod M$;
7. $\quad$ if $V_i \neq 0$ then $C := C \times T^{V_i} \bmod M$;
8. return C;

end algorithm.

In adaptive methods [7] the computation depends on the input data, such as the exponent E. M-ary methods and window methods pre-compute powers of all possible partitions, not taking into account that the partitions of the actual exponent may or may not include all possible partitions. Thus, the number of modular multiplications in the pre-processing step can be reduced if partitions of E do not contain all possible ones.

Let $\wp(E)$ be the list of partitions obtained from the binary representation of E. Assume that the list of partition is non-redundant and ordered according to the ascending decimal value of the partitions contained in the expansion of E. Recall that V_i and L_i are the decimal value and the number of digits of partition P_i. The generic algorithm for describing the computation of $T^E \bmod M$ using the window methods is given in Algorithm 4.

In Algorithm 2 and Algorithm 3, it is clear how to perform the pre-computation indicated in lines 2–3. For instance, let E = 1011001101111000. The pre-processing

step of the 4-ary method needs 14 modular multiplications ($T \rightarrow T \times T = T^2 \rightarrow T \times T^2 = T^3 \rightarrow \ \rightarrow T \times T^{14} = T^{15}$) and that of the maximum 4-digit sliding window method needs only 8 modular multiplications ($T \rightarrow T \times T = T^2 \rightarrow T \times T^2 = T^3 \rightarrow T^3 \times T^2 = T^5 \rightarrow T^5 \times T^2 = T^7 \rightarrow \ \rightarrow T^{13} \times T^2 = T^{15}$). However the adaptive 4-ary method would partition the exponent as $E = 1011\|0011\|0111\|1000$ and hence needs to pre-compute the powers T^3, T^7, T^8 and T^{11} while the method maximum 4-digit sliding window method would partition the exponent as $E = 1\|0\|11\|00\|11\|0\|1111\|000$ and therefore needs to pre-compute the powers T^3 and T^{15}. The pre-computation of the powers needed by the adaptive 4-digit sliding window method may be done using 6 modular multiplications $T \rightarrow T \times T = T^2 \rightarrow T \times T^2 = T^3 \rightarrow T^2 \times T^2 = T^4 \rightarrow T^3 \times T^4 = T^7 \rightarrow T^7 \times T = T^8 \rightarrow T^8 \times T^3 = T^{11}$ while the pre-computation of those powers necessary to apply the adaptive sliding window may be accomplished using 5 modular multiplications $T \rightarrow T \times T = T^2 \rightarrow T \times T^2 = T^3 \rightarrow T^2 \times T^3 = T^5 \rightarrow T^5 \times T^5 = T^{10} \rightarrow T^5 \times T^{10} = T^{15}$. Note that Algorithm 4 does not suggest how to compute the powers (lines 2–3) needed to use the adaptive window methods. Finding the best way to compute them is a NP-hard problem [4], [7].

Algorithm 4. AdaptiveWindowMethod(T, M, E)
1. Partition E using the given strategy;
2. for each partition $P_i \in \wp$ do
3. Compute $T^{V_i} \bmod M$;
4. $C := T^{V_{p-1}} \bmod M$;
5. for $i := p-2$ downto 0 do
6. $C := C^{L_i} \bmod M$;
7. if $V_i \neq 0$ then $C := C \times T^{V_i} \bmod M$;
8. return C;

end algorithm.

3 Addition Chains and Sequences

An *addition chain* of length l for an positive integer N is a list of positive integers $(E_1, E_2, \ldots, E_l)$ such that $E_1 = 1$, $E_l = N$ and $E_k = E_i + E_j$, $0 \leq i \leq j < k \leq l$. Finding a minimal addition chain for a given positive integer is an NP-hard problem. It is clear that a short addition chain for exponent E gives a fast algorithm to compute $T^E \bmod M$ as we have if $E_k = E_i + E_j$ then $T^{E_k} = T^{E_i} \times T^{E_j}$. The adaptive window methods described earlier use a near optimal addition chain to compute $T^E \bmod M$. However these methods do not prescribe how to perform the pre-processing step (Line 3 of Algorithm 4). In the following we show how to perform this step with minimal number of modular multiplications.

3.1 Addition sequences

There is a generalisation of the concept of addition chains, which can be used to formalise the problem of finding a minimal sequence of powers that should be computed in the pre-processing step of the adaptive window method.

An *addition sequence* for the list of positive integers $V_1, V_2, \ldots, V_p$ such that $V_1 < V_2 < \ldots < V_p$ is an addition chain for integer V_p that includes all the integers $V_1, V_2, \ldots, V_p$. The length of an addition sequence is the numbers of integers that constitute the chain. An addition sequence for a list of positive integers $V_1, V_2, \ldots, V_p$ will be denoted by $\xi(V_1, V_2, \ldots, V_p)$.

Hence, to optimise the number of modular required multiplications in the pre-processing step of the adaptive window methods for computing $T^E \bmod M$, we need to find an addition sequence of minimal length (or simply minimal addition sequence) for the values of the partitions included in the non-redundant ordered list $\wp(E)$. This is an NP-hard problem and we use genetic algorithm to solve it. Our method showed to be very effective for large window size. General principles of genetic algorithms are explained in the next section.

3.2 Brun's algorithm

Now we describe briefly, Brun's algorithm [1] to compute addition sequences. The algorithm is a generalisation of the continued fraction algorithm [1]. Assume that we need to compute the addition sequence $\xi(V_1, V_2, \ldots, V_p)$. Let $Q = \lfloor \frac{V_p}{V_{p-1}} \rfloor$ and let $\chi(Q)$ be the addition chain for Q using the binary method (i.e. that used in Algorithm 2 with l = 1). Let $R = V_p - Q \times V_{p-1}$. By induction we can construct an addition sequence $\xi(V_1, V_2, \ldots, R, \ldots, V_{p-1})$, then obtain:

$$\begin{aligned} \xi(S) = \quad & \xi(V_1, V_2, \ldots, R, \ldots, V_{p-1}) \cup \\ & V_{p-1} \times \chi(Q) \setminus \{1\} \cup \{V_p\}, \\ S = \quad & V_1, V_2, \ldots, V_p \end{aligned} \tag{1}$$

4 Ant Systems and Algorithms

Ant systems can be viewed as multi-agent systems [3] that use a shared memory through which they communicate and a local memory to bookkeep the locally reached problem solution. Fig. 1. depicts the overall structure of an system, wherein A_i and LM_i represent the $i^{th.}$ agent of the ant system and its local memory respectively. Mainly, the shared memory (*SM*) holds the pheromone information while the local memory LM_i keeps the solution (possibly partial) that agent A_i reached so far.

The behaviour of an artificial ant colony is summarised in Algorithm 5, wherein N, C, SM are the number of artificial ant that form the colony, the characteristics of the expected solution and the shared memory used by the artificial ants to store pheromone information respectively.

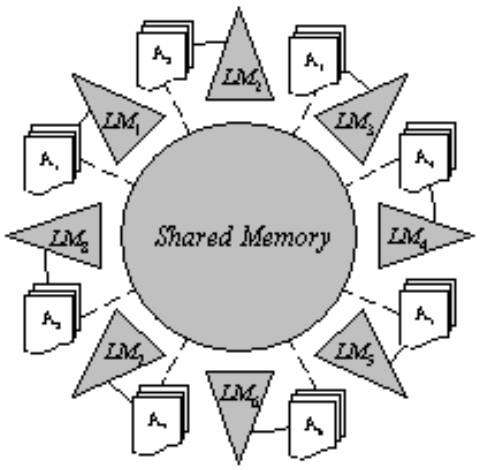


Figure 1: Multi-agent system architecture

The first step consists of activating N distinct artificial ants that should work in simultaneously. Every time an ant conclude its search, the shared memory is updated with an amount of pheromone, which should be proportional to the quality of the reached solution. This called *global* pheromone update. When the solution yield by an ant's work is suitable (i.e. fits characteristics C) then all the active ants are stopped. Otherwise, the process is iterated until an adequate solution is encountered.

Algorithm 5. ArtificialAntColony(N, C)
1. Initialise SM with initial pheromone;
2. do
3. for $i := 1$ to N
4. Start ArtificialAnt(A_i, LM_i);
5. $Active := Active \cup \{A_i\}$;
6. do
7. Update SM (pheromone evaporation);
8. when an ant (say A_i) halts do
9. $Active := Active \setminus \{A_i\}$;
10. Φ := Pheromone(LM_i);
11. Update SM (global pheromone Φ);
12. S := ExtractSolution(LM_i);
13. until Fitness$(S) = C$ or $Active = \emptyset$;
14. while $Active \neq \emptyset$ do
15. Stop ant $A_i \mid A_i \in Active$;
16. $Active := Active \setminus \{A_i\}$;
17. until Fitness$(S) = C$;
18. return S;

end.

The behaviour of an artificial ant is described in Algorithm 6, wherein A_i and LM_i represent the ant identifier and the ant local memory, in which it stores the solution computed so far. First, the ant computes the probabilities that it uses to select the next state to move to. The computation depends on the solution built so far, the problem constraints as well as some heuristics [2], [6]. Thereafter, the ant updates the solution stored in its local memory, deposits some *local* pheromone into the shared memory then moves to the chosen state. This process is iterated until complete problem solution is yielded.

Algorithm 6. ArtificialAnt(A_i, LM_i)
1. Initialise LM_i;
2. do
3. P := TransitionProbabilities(LM_i);
4. $NextState$:= StateDecision(LM_i, P);
5. Update LM_i;
6. Update SM with local pheromone;
7. $CurrentState := NextState$);
8. until $CurrentState := TargetState$;
9. Halt A_i;

end.

5 Chain Sequence Minimisation Using Ant System

In this section, we concentrate on the specialisation of the ant system of Algorithm 4 and Algorithm 5 to the addition sequence minimisation problem. For this purpose, we describe how the shared and local memories are represented. We then detail the function that yields the solution (possibly partial) characteristics. Thereafter, we define the amount of pheromone to be deposited with respect to the solution obtained so far. Finally, we show how to compute the necessary probabilities and make the adequate decision towards a shorter addition sequence for the considered the sequence $(V_1, V_2, \ldots, V_p)$.

5.1 The Ant System Shared Memory

The ant system shared memory is a two-dimension array. If the last exponent in the sequence is V_p then the array should V_p rows. The number of columns depends on the row. It can be computed as in Eq. 2, wherein NC_i denotes the number of columns in row i.

$$NC_i = \begin{cases} 2^{i-1} - i + 1 & \text{if } 2^{i-1} < V_p \\ 1 & \text{if } i = V_p \\ V_p - i + 3 & \text{otherwise} \end{cases} \quad (2)$$

An entry $SM_{i,j}$ of the shared memory holds the pheromone deposited by ants that used exponent $i+j$ as the i th. member in the built addition sequence. Note that $1 \leq i \leq V_p$ and for row i, $0 \leq j \leq NC_i$. Fig. 2 gives an example of the shared memory for exponent 17. In this example, a table entry is set to show the exponent corresponding to it. The exponent $E_{i,j}$ corresponding to entry $SM_{i,j}$ should be obtainable from exponents of previous rows. Eq. 3 formalises such a requirement.

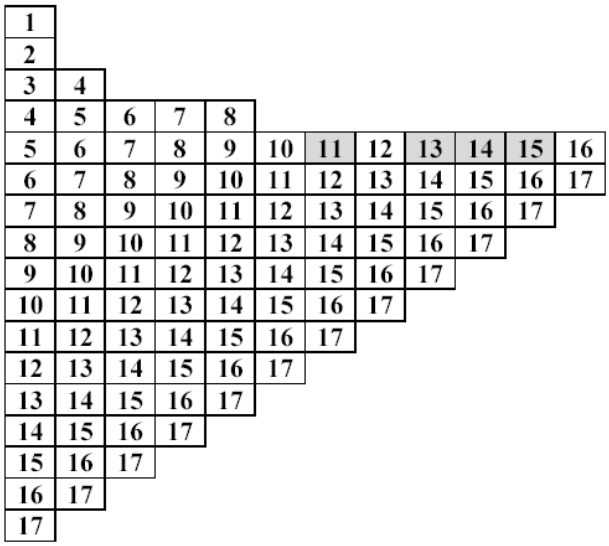


Figure 2: Example of shared memory content for $V_p = 17$

$$E_{i,j} = E_{k_1,l_1} + E_{k_2,k_2} | \quad \begin{array}{l} 1 \leq k_1, k_2 < i, \\ 0 \leq l_1, l_2 \leq j, \\ k_1 = k_2 \Longleftrightarrow l_1 = l_2 \end{array} \tag{3}$$

Note that, in Fig. 2, the exponents in the shaded entries are not valid exponents as for instance exponent 7 of row 4 can is not obtainable from the sum of two previous different stages, as described in Eq. 3. The computational process that allows us to avoid these exponents is of very high cost. In order to avoid using these few exponents, we will penalise those ants that use them and hopefully, the solutions built by the ants will be almost all valid addition chains. Furthermore, note that for a valid solution need also to contain all the exponents of the sequence i.e., $V_1, V_2, \ldots, V_{p-1}, V_p$.

5.2 The Ant Local Memory

In an ant system, each ant is endowed a local memory that allows it to store the solution or the part of it that was built so far. This local memory is divided into two parts: the first part represents the (partial) addition sequence found by the ant so far and consists of a one-dimension array of V_p entries; the second part holds the *characteristic* of the solution. It represents the solution fitness i.e., its length. The details of how to compute the fitness of a possibly partial addition sequence are given in the next section. Fig. 3 shows six different examples of an ant local memory for sequence (5, 7, 11). Fig. 3(a) represents addition sequence (1, 2, 4, 5, 7, 11), which is a valid and complete solution of fitness 5. Fig. 3(b) depicts addition sequence (1, 2, 3, 5, 7, 10, 11), which is also a valid and complete solution but of fitness 6. Fig. 3(c) represents partial addition sequence (1, 2, 4, 5), which is a valid and but incomplete solution as it does not include exponent 7 and 11 and the last exponent

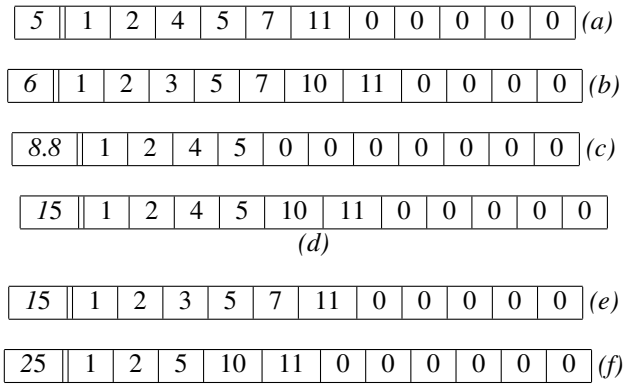


Figure 3: Example of an ant local memory

is smaller than both 7 and 11. The corresponding fitness is 8.8. Fig. 3(d) consists of non-valid addition sequence (1, 2, 4, 5, 10, 11) as 7 is not included. The corresponding fitness is 15. Fig. 3(e) represents also non-valid addition sequence (1, 2, 3, 5, 7, 11) as 11 is not a sum two previous exponents in the sequence. Its fitness is also 15. Finally, Fig. 3(f) represents also non-valid addition sequence (1, 2, 5, 10, 11) as 5 is not a sum two previous and mandatory exponent 7 is not in the addition sequence. exponents in the sequence. Its fitness is also 25. In next section, we explain how the fitness of a solution is computed.

5.3 Addition Sequence Characteristics

The fitness evaluation of an addition sequence is performed with respect to three aspects: *(a)* how much it adheres to the definition (see Section 3), i.e. how many of its members cannot be obtained summing up two previous members of the sequence; *(b)* how far the it is reduced, i.e. what is the length of the chain; *(c)* how many of the mandatory exponents do not appear in the sequence. Eq. 4 shows how to compute the fitness f of solution $\alpha = (E_1, E_2, \ldots, E_n, 0, \ldots, 0)$ regarding mandatory exponents $\sigma = V_1, V_2, \ldots, V_p$.

$$\begin{array}{ll} f(S,A) = & \frac{V_p \times (n-1)}{E_n} + \psi \times (\eta_1 + \eta_2) \\ \sigma = & V_1, V_2, \ldots, V_p \\ \alpha = & V_1, V_2, \ldots, V_p \end{array} \tag{4}$$

wherein ψ is a penalty, η_1 represents the number of E_i, $3 \leq i \leq n$ in the addition sequence that verify the predicate below:

$$\forall j, k \mid \ 1 \leq j, k < i, E_i \neq E_j + E_k \tag{5}$$

and η_2 represents the number of mandatory exponents V_i, $1 \leq i \leq p$ that verify the predicate below:

$$V_i \leq E_n \Longrightarrow \forall j \mid \ 1 \leq j \leq n, E_j \neq V_i \tag{6}$$

For a valid complete addition sequence, the fitness coincides with its length, which is the number of multiplica-

tions that are required to compute the exponentiation using the sequence. For a valid but incomplete addition sequence, the fitness consists of its *relative* length. It takes into account the distance between last mandatory exponent V_p and the last exponent in the partial addition sequence. Furthermore, for every mandatory exponent that is smaller than the last member of the sequence which is not part of it, a penalty is added to the sequence fitness. Note that valid incomplete sequences may have the same fitness of some other valid and complete ones. For instance, addition sequence (1, 2, 3, 6, 8) and (1, 2, 3, 6) for exponent mandatory exponents (3, 6, 8) have the same fitness 4.

For an invalid addition sequences, a penalty, which should be larger than V_p, is introduced into the fitness value for each exponent for which one cannot find two (may be equal) members of the sequence whose sum is equal to the exponent in question or two distinct previous members of the chain whose difference is equal to the considered exponent. Furthermore, a penalty is added to the fitness of a addition sequence whenever the a mandatory exponent is not part of it. The penalty used in the examples of Fig. 3 is 10.

5.4 Pheromone Trail and State Transition Function

There are three situations wherein the pheromone trail is updated: *(a)* when an ant chooses to use exponent $F = i+j$ as the *i*th. member in its solution, the shared memory cell $SM_{i,j}$ is incremented with a constant value of pheromone $\Delta\phi$, as in the first assignment of Eq. 7; *(b)* when an ant halts because it reached a complete solution, say $\alpha = (E_1, E_2, \ldots, E_n)$ for mandatory exponent sequence σ, all the shared memory cells $SM_{i,j}$ such that $i+j = E_i$ are incremented with pheromone value of $1/Fitness(\sigma, \alpha)$, as in the second Eq. 7. Note that the better is the reached solution, the higher is the amount of pheromone deposited in the shared memory cells that correspond to the addition sequence members. *(iii)* The pheromone deposited should evaporate. Periodically, the pheromone amount stored in $SM_{i,j}$ is decremented in an exponential manner [6] as in the third assignment of Eq. 7.

$$\begin{aligned} SM_{i,j} &:= SM_{i,j} + \Delta\phi, \text{ whenever } E_i = i + j \\ SM_{i,j} &:= SM_{i,j} + 1/f(\sigma, \alpha), \forall i, j \mid i + j = E_i \\ SM_{i,j} &:= (1 - \rho)SM_{i,j} \mid \rho \in (0, 1], \text{ periodically} \end{aligned} \quad (7)$$

An ant, say A that has constructed partial addition sequence $(E_1, E_2, \ldots, E_i, 0, \ldots, 0)$ for exponent sequence $(V_1, V_2, \ldots, V_p)$, is said to be in *step i*. In step $i + 1$, it may choose exponent E_{i+1} $E_i + 1$, $E_i + 2$, …, $2E_i$, if $2E_i \leq V_p$. That is, ant A may choose one of the exponents that are associated with the shared memory cells SM_{i+1,E_i-i}, SM_{i+1,E_i-i+1}, …, $SM_{i+1,2E_i-i-1}$. Otherwise (i.e. if $2E_i > V_p$), it may only select from exponents $E_i + 1$, $E_i + 2$, …, $E + 2$. In this case, ant A may choose one of the exponent associated with $SM_{i+1,E_i-i}, SM_{i+1,E_i-i+1}, \ldots, SM_{i+1,E-i+1}$. Furthermore, ant A chooses the new exponent E_{i+1} with the probability expressed through Eq. 8 below.

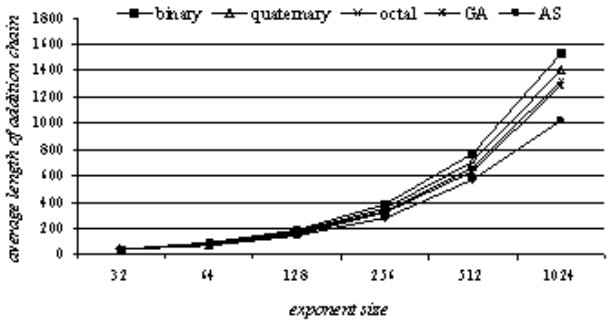


Figure 4: Comparison of the average length of the addition chains

$$P_{i,j} = \begin{cases} \dfrac{SM_{i+1,j}}{\max\limits_{k=E_i-i}^{2E_i-i-1} SM_{i+1,k}} & \text{if } 2E_i \leq E \;\&\; j \in [E_i - i, 2E_i - i - 1] \\ \dfrac{SM_{i+1,j}}{\max\limits_{k=E_i-i}^{E-i-1} SM_{i+1,k}} & \text{if } 2E_i > E \;\&\; j \in [E_i - i, E - i - 1] \\ 0 & \text{otherwise} \end{cases} \quad (8)$$

6 Performance Comparison

The ant system described in Algorithm 5 and Algorithm 6 was implemented using Java as a multi-threaded ant system. Each ant was simulated by a thread that implements the artificial ant computation of Algorithm 4. A Pentium IV-HT™ of a operation frequency of 1GH and RAM size of 2GB was used to run the ant system and obtain the performance results.

We compared the performance of m-ary methods, the Brun's algorithm, genetic algorithms and ant system-based methods. The obtained addition chains are given in Table 1.

The average lengths of the addition sequences for different exponent sequences obtained using these methods are given in Table 2. The exponent size is that of its binary representation (i.e. number of bits). The ant system-based method always outperforms all the others, including the genetic algorithm-based method [7]. The chart of Fig. 4 shows the relation between the average length of the obtained addition sequences.

Table 1: The addition sequences yield for $\xi(5,9,23)$, $\xi(9,27,55)$ and $\xi(5,7,95)$ respectively

Method	Addition sequence	$\#\times$
5-ary	(1,2,3,. . .,30,31)	30
5-window	(1,2,3,5,7,9,11,. . .,31)	16
Brun's	(1,2,4,5,9,18,23)	6
GAs	(1,2,4,5,9,18,23)	6
Ant system	(1,2,4,5,9,14,23)	6
6-ary	(1,2,3,. . .,. . .,63)	62
6-window	(1,2,3,5,7,. . .,63)	31
Brun's	(1,2,3,6,9,18,27,54,55)	8
GAs	(1,2,4,8,9,18,27,28,55)	8
Ant system	(1,2,4,5,9,18,27,54,55)	8
7-ary	(1,2,3,4,5,6,7,. . .,95)	94
7-window	(1,2,3,5,7,. . .,95)	43
Brun's	(1,2,4,5,7,14,21,42,84,91,95)	10
GAs	(1,2,3,5,7,10,20,30,35,65,95)	10
Ant system	(1,2,4,5,7,14,19,38,76,95)	9
7-ary	(1,2,3,4,5,6,7,. . .,95)	94
7-window	(1,2,3,5,7,. . .,95)	43
Brun's	(1,2,4,5,7,14,21,42,84,91,95)	10
GAs	(1,2,3,5,7,10,20,30,35,65,95)	10
Ant system	(1,2,4,5,7,14,19,38,76,95)	9

Table 2: Average length of addition sequence for Brun's algorithm (BA), genetic algorithms (GA) and ant system (AS)

$\lvert V_p\rvert$	BA	GA	AS
32	41	42	45
64	84	85	86
128	169	170	168
256	340	341	331
512	681	682	658
1024	1364	1365	1313

7 Conclusion

In this paper we applied the methodology of ant colony to the addition chain minimisation problem. Namely, we described how the shared and local memories are represented. We detailed the function that computes the solution fitness. We defined the amount of pheromone to be deposited with respect to the solution obtained by an ant. We showed how to compute the necessary probabilities and make the adequate decision towards a good addition chain for the considered exponent.

Furthermore, we implemented the ant system described using multi-threading (each ant of the system was implemented by a thread). We compared the results obtained by the ant system to those of *m*-ary methods (binary, quaternary and octal methods). Taking advantage of the a previous work on evolving minimal addition chains with genetic algorithm, we also compared the obtained results to those obtained by the genetic algorithm. The ant system always finds a shorter addition chain and gain increases with the size of the exponents.

References

[1] Rivest, R., Shamir, A. and Adleman, L., A method for Obtaining Digital Signature and Public-Key Cryptosystems, Communications of the ACM, 21:120-126, 1978.

[2] Dorigo, M. and Gambardella, L.M., Ant Colony: a Cooperative Learning Approach to the Travelling Salesman Problem, IEEE Transaction on Evolutionary Computation, Vol. 1, No. 1, pp. 53-66, 1997.

[3] Feber, J., Multi-Agent Systems: an Introduction to Distributed Artificial Intelligence, Addison-Wesley, 1995.

[4] Downing, P. Leong B. and Sthi, R., Computing Sequences with Addition Chains, SIAM Journal on Computing, vol. 10, No. 3, pp. 638-646, 1981.

[5] Nedjah, N., Mourelle, L.M., Efficient Parallel Modular Exponentiation Algorithm, Second International Conference on Information systems, ADVIS'2002, Izmir, Turkey, Lecture Notes in Computer Science, Springer-Verlag, vol. 2457, pp. 405-414, 2002.

[6] Stutzle, T. and Dorigo, M., ACO Algorithms for the Travelling Salesman Problems, Evolutionary Algorithms in Engineering and Computer Science, John-Wiley & Sons, 1999.

[7] Nedjah, N. and Mourelle, L.M., Minimal addition-subtraction chains using genetic algorithms, Proceedings of the Second International Conference on Information Systems, Izmir, Turkey, Lecture Notes in Computer Science, Springer-Verlag, vol. 2457, pp. 303-313, 2002.

Max Min Ant System and Capacitated *p*-Medians: Extensions and Improved Solutions

Fabrício Olivetti de França and Fernando J. Von Zuben
LBiC/DCA/FEEC
State University of Campinas (Unicamp)
PO Box 6101, 13083-852 – Campinas/SP, Brazil

Leandro Nunes de Castro
Research and Graduate Program on Informatics
Catholic University of Santos (UniSantos)
R. Dr. Carvalho de Mendonça, 144
11070-906, Santos/SP, Brazil

Keywords: Ant colony optimization, Capacitated p-medians, Clustering problems

Received: September 24, 2004

This work introduces a modified MAX MIN Ant System (MMAS) designed to solve the Capacitated p*-Medians Problem (CPMP). It presents the most relevant steps towards the implementation of an MMAS to solve the CPMP, including some improvements on the original MMAS algorithm, such as the use of a density model in the information heuristics and a local search adapted from the uncapacitated* p*-medians problem. Extensions of a recently proposed updating rule for the pheromone level, aiming at improving the MMAS ability to deal with large-scale instances, are also presented and discussed. Some simulations are performed using instances available from the literature, and well-known heuristics are employed for benchmarking.*

Povzetek: Predstavljene so izboljšave algoritma MAX MIN na osnovi kolonij mravelj za reševanje problema CPMP.

1 Introduction

The capacitated *p*-medians problem (CPMP), also known as capacitated clustering problem, is a combinatorial programming task that can be described as follows: given a graph with *n* vertices (clients), find *p* centers (medians) and assign the other vertices to them minimizing the total distance covered, limited to a capacity restriction. This problem is a special case of the "capacitated plant location problem with single source constraints" and many other combinatorial problems as pointed in Osman and Christofides [1]. As such, the CPMP was proved to be NP-complete in Garey and Johnson [2]. Its practical use varies from industrial and commercial planning to every clustering related problem, like data mining, pattern recognition, vehicle routing and many others.

Ant Systems (AS) were first proposed in Dorigo [3] as an attempt to use the ant foraging behavior as a source of inspiration for the development of new search and optimization techniques. By using the pheromone trail as a reinforcement signal for the choice of which path to follow, ants tend to find "minimal" routes from the nest to the food source. The system is based on the fact that ants, while foraging, deposit a chemical substance, known as pheromone, on the path they use to go from the food source to the nest. The standard system was later extended in Dorigo and Di Caro [4], giving rise to the so-called Max Min Ant System (MMAS). The main purpose of the max-min version is to improve the search capability of the standard algorithm by combining exploitation with exploration of the search space, and by imposing bounds to the pheromone level, thus helping to avoid stagnation.

This paper is an extension of the work initiated in de França et al. [5], with additional contributions: a thorough analysis and explanation of the proposed operators, and a broader set of experiments. Essentially, the innovative aspects of the approach are twofold: (*i*) adaptation of the MMAS algorithm to deal with a problem not previously conceived by means of an ant-based formalism; and (*ii*) proposition of several modifications to the MMAS algorithm so as to improve its performance when dealing with large instances of combinatorial optimization problems. In practical terms, the ant system will incorporate a local search procedure for the CPMP, a new updating rule for the pheromone level, and a stagnation control mechanism.

The paper is organized as follows. Section 2 provides a mathematical formulation of the CPMP problem and the General Assignment Problem (GAP) that results when the medians are already specified. In Section 3, the basic Ant System algorithm together with its Max Min version, MMAS, are reviewed. Section 4 emphasizes the contributions of this work. It describes the proposed enhancements of MMAS, leading to the improved MMAS, called here IMMAS, and how to apply ant-based algorithms to the capacitated *p*-medians problem. The proposed algorithm is evaluated in Section 5, and its performance is compared with that of other works from the literature. The paper is concluded in Section 6 with a discussion about the formal and methodological contri-

butions and a description of several avenues for further investigation.

2 Mathematical Formulation of the Capacitated p-Medians Problem

This section provides a mathematical formulation of the capacitated *p*-medians problem as a constrained optimization problem: the total distance from the medians to the clients has to be minimized, constrained by the demands of clients and capacities of medians.

On a complete graph, given n nodes with predefined capacities and demands, the goal is to choose p nodes ($p < n$) as capacitated medians and to attribute each one of the remaining $(n - p)$ nodes, denoted clients, to one of the chosen medians, so that the capacity of each median is not violated by the cumulated demand, and the sum of the distances from each client to the corresponding median is minimal. Every node is a candidate to become a median, and the solution will consider demand and capacity of medians, and only demand of clients.

Defining an $n \times n$ matrix $\mathbf{X}$, with components $x_{ij} \in \{0,1\}$, $i,j = 1,...,n$, and an n-dimensional vector $\mathbf{y}$, with components $y_j \in \{0,1\}$, $j = 1,...,n$, the following associations are imposed:

$$x_{ij} = \begin{cases} 1, \text{if node } i \text{ is allocated to median } j \\ 0, \text{otherwise} \end{cases}$$

$$y_j = \begin{cases} 1, \text{if node } j \text{ is a median} \\ 0, \text{otherwise} \end{cases}$$

The CPMP formulation as an integer-programming problem can, thus, be given as follows:

$$\min_{\mathbf{X},y} f(\mathbf{X}) \equiv \min_{\mathbf{X},y} \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} x_{ij} \cdot \quad (1)$$

subject to,

$$\begin{cases} \sum_{i=1}^{n} x_{ij} = 1, j = 1,...,n \\ x_{ij} \le y_j, i, j = 1,2,...,n \\ \sum_{j=1}^{n} y_j = p \\ \sum_{i=1}^{n} x_{ij}.a_i \le c_j, \text{for } j \text{ such that } y_j = 1 \end{cases} \quad (2)$$

where:

n = number of nodes in the graph
a_i = demand of node i
c_j = capacity of median j
d_{ij} = distance between nodes i and j
p = number of medians to be allocated

After all p medians are chosen, the CPMP becomes a Generalized Assignment Problem (GAP); that is, given a set of medians, allocate a set of clients to those medians so as to minimize Eq. (3):

$$\min_{\mathbf{X}} \sum_{i=1}^{n} \sum_{j=1}^{p} d_{ij} x_{ij} \cdot \quad (3)$$

where d_{ij} is the cost of allocating client i to median j, n is the number of nodes, and p is the number of medians, subject to the capacity of the respective medians and client demands:

$$\begin{cases} \sum_{i=1}^{p} x_{ij} = 1, j = 1,..., p, \\ \sum_{i=1}^{n} x_{ij} \cdot a_i \le c_j, j = 1,..., p \\ x_{ij} \in \{0,1\}, i = 1,..,m; j = 1,.., p \end{cases} \quad (4)$$

where a_i is the demand of client i, and c_j is the capacity of median j.

3 Ant System and Max-Min Ant System

The ant system (AS) [3] was the first ant-based algorithm applied to solve combinatorial optimization problems. More than one decade after it was introduced, several different versions, improvements and applications have been presented (c.f. Dorigo and Di Caro [4], Dorigo and Stützle [6], de Castro and Von Zuben [7]). This section briefly reviews the original proposal together with one of its most popular variants, the Max Min Ant System (MMAS).

3.1 Ant System

The basic AS [3] is conceptually simple, as described in Algorithm 1.

```
Function AS()
    While it < max_it do,
        For each ant do,
          build_solution();
          update_pheromone();
        Endfor
    Endwhile
End
```

Algorithm 1: Pseudocode for the basic ant system (AS).

In Algorithm 1, procedure build_solution() builds a solution to a problem based on a pheromone trail and on optional information heuristics. Each ant k traverses one node per iteration step t and, at each edge, the local information about its pheromone level, τ_{ij}, is used by the ant such that it can probabilistically decide the next node to move to, according to the following rule:

$$p_{ij}^{k}(t) = \begin{cases} \dfrac{[\tau_{ij}(t)]}{\sum_{j \in J^k}[\tau_{ij}(t)]} & \text{if } j \in J^k \\ 0 & \text{otherwise} \end{cases}, \qquad (5)$$

where $\tau_{ij}(t)$ is the pheromone level of edge (i,j), and J^k is the list of nodes yet to be visited by ant k.

While traversing an edge (i,j), ant k deposits some pheromone on it – procedure update_pheromone() – and the pheromone level of edge (i,j) is updated according to Eq. (6).

$$\tau_{ij} \leftarrow \rho.\tau_{ij} + \Delta\tau_{ij}, \qquad (6)$$

where $\rho \in (0,1]$ is the pheromone decay rate, and $\Delta\tau_{ij}$ is the increment in the pheromone level. In minimization problems, the pheromone increment is given by

$$\Delta\tau_{ij} = \begin{cases} 1/f(S), & \text{if } (i,j) \in S \\ 0, & \text{otherwise} \end{cases}, \qquad (7)$$

where S is the solution used to update the trail, and $f(S)$ is a function that reflects the quality of a solution, i.e., the lower the value the better the quality assuming a minimization problem is being solved.

In our proposal, the pheromone is represented as a vector, instead of as a bi-dimensional matrix as in the classical AS. This is because the algorithm to be described here works by assigning pheromone to vertices and not to edges, as will be further discussed in Section 4.

3.2 Max Min Ant System (MMAS)

An important improvement to the Ant System, called Max Min Ant System (MMAS), was introduced in Stützle and Hoos [8]. In this implementation, the pheromone trail is updated only on the global best and/or local best solutions, instead of on solutions created by every ant. This promotes a better exploitation of the search space, as it favors the solutions in the neighborhood of the global and local bests. Another improvement is the inclusion of upper and lower bounds to the pheromone level (τ_{max} and τ_{min}), thus helping to avoid stagnation. Initially all trail is set to the upper bound in order to favor exploration. As defined in Stützle and Hoos [8], and in Stützle and Dorigo [9], the upper bound is usually chosen to be the maximum value the pheromone can reach at the final iterations. Following from Eq. (6), the maximum value at a given iteration t is:

$$\tau_i(t) = \sum_{j=1}^{t} \rho^{t-j} \cdot \frac{1}{F_{opt}} + \rho^t.\tau_0 . \qquad (8)$$

where F_{opt} is the optimal solution, ρ is the pheromone decay rate, and τ_0 is the initial pheromone value. As $\rho < 1$, when t tends to infinity, the pheromone value is limited to

$$\tau_{max} = \frac{1}{1-\rho} \cdot \frac{1}{F_{opt}} . \qquad (9)$$

The problem with Eq. (9) is that the optimal solution F_{opt} is usually unknown. To circumvent this difficulty, F_{best}, that is, the fitness of the best solution found so far, is used in place of F_{opt}, as an approximation.

The lower bound is calculated so as to give a τ_{max}/τ_{min} ratio equal to $2n$ (twice the set of candidates to medians), so it is set to $\tau_{min} = \tau_{max}/2n$. On the one hand, this ratio must not be too high, because the probability of selecting a path with low pheromone level would become too small. On the other hand, if the ratio is too low, the probability of selecting a path with high pheromone level would be very close to the probability of selecting a path with low pheromone level.

4 MMAS Applied to the Capacitated p-Medians Problem

This section describes the general methodology used to apply the proposed modified Max Min Ant System to the CPMP [5]. In particular, it is described how the solutions are built by the algorithm, the use of a density-based information heuristics, a local search procedure, a new updating rule for the ant system, and a stagnation control mechanism.

4.1 Building Solutions

The construction of each solution to the CPMP using MMAS is made as follows. Using the probabilistic equation, Eq. (5), each ant sequentially chooses a set of p nodes to become medians among the n candidate nodes. Note that the pheromone level in our proposal is attributed to nodes and not edges. After the definition of the p nodes that will play the role of medians, each one of the remaining $n - p$ nodes has to be allocated to precisely one median, giving rise to the Generalized Assignment Problem (GAP) described in Section 2.

The resulting GAP will not be solved using ant system. Instead, a constructive heuristic to allocate clients to medians will be adopted. The method used here was proposed by Osman and Christofides [1] and works as summarized in Algorithm 2.

```
Function [x] = GAP(clients[ ],medians[ ],n,p)
    ordered_clients = sort_clients();
    For i = 1 to n do,
      ordered_medians = sort_medians(ordered_clients[i]);
      For j = 1 to p do,
        If (capacity(ordered_medians[j]) –
            demand(ordered_clients[i])) >= 0,
              x[ordered_clients[i]][ordered_medians[j]]=1;
        Endif
      Endfor
    Endfor
End
```

Algorithm 2: Constructive heuristics to allocate clients to medians.

Function sort_clients() generates a list with all the n clients in increasing order of distance to their corresponding nearest median. Then, the algorithm loops sequentially

through this list calling sort_medians() to each client, which generates a list with all the p medians in increasing order of distance to the current client. Given the ordered list of medians, the current client will be allocated to the first available median, i.e., the one for which the difference between median capacity and client demand is greater or equal to zero. The result is the matrix x_{ij} described in Section 2.

After these steps the solution is evaluated by means of Eq. (1). Then, one iteration of local search is performed as described in Section 4.3. This procedure is then repeated for each ant. For stagnation control, if the algorithm does not improve the solution for 30% (defined empirically) of the number of total iterations, all pheromone trails are restarted.

4.2 Information Heuristic (η)

In order to improve the solutions found by the constructive phase of the ant algorithm, an information heuristic which contains the quality of choosing each node as part of the solution is used. For this heuristic, some greedy information concerning the problem is often adopted, such as the distance among the current node to the others in traveling salesperson problems as shown in Dorigo [3], Dorigo and Di Caro [4], and Dorigo and Stützle [6]. Thus, Eq. (5) becomes Eq. (10):

$$p_i^k(t) = \begin{cases} \dfrac{[\tau_i(t)]^{\alpha}.[\eta_i]^{\beta}}{\sum_{l \in J^k}[\tau_l(t)]^{\alpha}.[\eta_l]^{\beta}} & \text{if } i \in J^k \\ 0 & \text{otherwise} \end{cases}, \qquad (10)$$

The parameters α and β are user-defined and control the relative weight of trail intensity $\tau_i(t)$ and information heuristic η_i.

In the case of CPMP, the information heuristic proposed here is a density model for this problem based on Ahmadi and Osman [10]. The idea is to calculate an optimistic density of a cluster if a given node was to be chosen as the median. The computation follows Algorithm 3.

```
Function [η] = density()
  For i = 1 to n do,
    ordered_nodes = sort_nodes(i);
    [all_nodes, sum_distance] = allocate(i,ordered_nodes);
    η_i = all_nodes / sum_distance;
  Endfor
End
```

Algorithm 3: Calculating the density of a cluster.

Function sort_nodes() sorts all nodes based on their distance to node i; and function allocate() assigns each node in ordered_nodes to i, until its capacity is reached, returning two outputs: (*i*) all_nodes: the number of allocated nodes; and (*ii*) sum_distance: the summation of the distance between each allocated node and node i. Although this is a reasonable measure of the potential of a node as a candidate to become a median, it does not always imply the most appropriate scenario. Given that information heuristic provides just an approximated indication of the best candidates to become medians, parameters α and β are set so as to emphasize pheromone instead of the heuristic information, as will be observed in the experiments described in Section 5. This is opposed to the approach usually adopted for tackling the TSP problem, in which the best results found are given more importance.

4.3 A Local Search Procedure for the CPMP

The local search heuristic is a first improvement approach for the MMAS algorithm. Basically it consists of changing a client into a median and this median into a client seeking an improvement of the objective function.

To define the search neighborhood, an approach based on the uncapacitated p-medians problem proposed in Resend and Werneck [11], and in Teitz and Bart [12] was adopted. This approach consists of an optimistic function to calculate a *profit*, P, obtained by changing a client into a median and determining which median to remove in this case. Initially, two vectors d_1 and d_2, containing the first and second closest median to each client, are calculated. Then, the profit associated with each possible interchange between a client and a median obeys the following equation:

$$P(f_i, f_r) = \sum_{u:d_1(u) \neq f_r} \max\left(0, [d_1(u) - d(u, f_i)]\right) - \sum_{u:d_1(u)=f_r} \left(\min(d_2(u), d(u, f_i)) - d_1(u)\right), \qquad (11)$$

where M is the set of medians, $f_i \notin M$ and $f_r \in M$ is applied, f_i is the node chosen to enter the solution as a median, f_r is the node chosen to leave it, $d_1(u)$ is the distance of node u to its nearest median, $d_2(u)$ is the distance of node u to its second nearest median, and $d(x_1,x_2)$ is the Euclidean distance between x_1 and x_2.

Figure 1 provides a general overview of the local search procedure. The larger circles represent candidates to medians, and the smaller ones represent the clients.

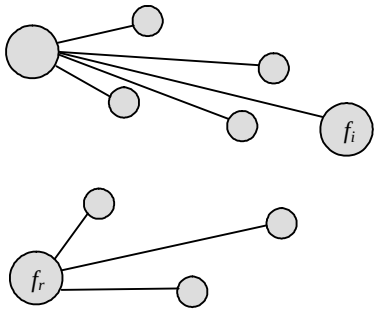


Figure 1: Overview of the local search procedure. Solid lines connect clients to medians, f_i is the candidate to enter the solution as a median, and f_r is the median that will leave the solution.

The first term of the right hand side of Eq. (11), detailed in Figure 2, refers to all clients that do not belong to the median candidate to leave the solution, and takes into account two possibilities: (i) the new median is nearer to the client than its previous nearest median; or (ii) the new median is farther to the client than its previous nearest median. When the former holds, allocating this client to the new median will reduce the total value of the objective function, increasing the *profit* in proportion to $d_1(u) - d(u,f_i)$. Otherwise, no change occurs.

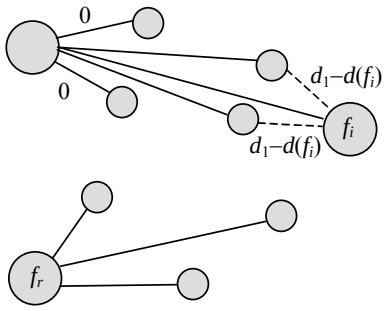


Figure 2: First term of the right hand side of Eq. (11). Clients not allocated to f_r that will profit from the insertion of f_i. Solid lines connect each client to a median, dotted lines represent the new connections, f_i is the candidate to enter the solution as a median, and f_r is the median that will leave the solution.

In the second term of the right hand side of Eq. (11), those clients that will lose their nearest median and will be allocated to a new one are taken into account (Figure 3). There are also two possibilities in this case: (i) the nearest median becomes the new median; or (ii) the client is allocated to its second nearest median. In the first case, the difference on the profit will be $d(u,f_i) - d_1(u)$, and can make the total distance larger or smaller. In the second case, the profit function will have a decrease proportional to $d_2(u) - d_1(u)$.

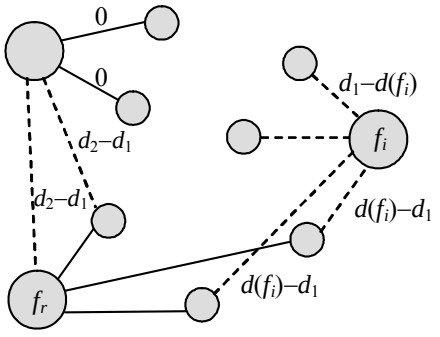


Figure 3: Second term of the right hand side of Eq. (11). Clients allocated to f_r that will profit or not from the exit of f_r. Solid lines connect each client to a median, dotted lines represent the new connections, f_i is the candidate to enter the solution as a median, and f_r is the median that will leave the solution.

After this procedure has finished, two similar First Improvement Local Search procedures are performed, but regarding the clients instead of medians. The first one (Figure 4(a)) consists of interchanging two clients of different medians whenever it is profitable, and after that, recalculating the medians taking the best point inside each cluster. The second type (Figure 4(b)) is the same as the previous one, but two clients of one median are interchanged with one client from another.

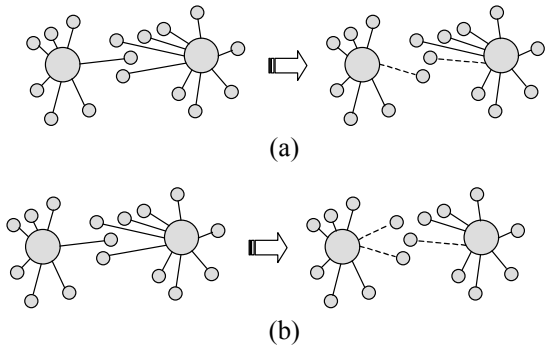


Figure 4: First Improvement Local Search procedures. (a) One movement of 1-interchange GAP local search. One client of the first cluster interchanges with a client from the other one (dotted lines). (b) One movement of 2-interchange GAP local search. One client of the first cluster interchanges with two clients from the other one (dotted lines).

A number greater than 2 for the λ-interchange algorithm is computationally expensive and will hardly represent significant benefits to the final results, because 1- and 2-interchange, when tried several times, can eventually perform the job of a λ-interchange for higher values of λ.

4.4 A New Updating Rule for AS

A well-known problem with the AS is that of scaling the objective function to update the pheromone trail. If not appropriately done, the performance of the algorithm tends to be unsatisfactory for large instances of the problem. To propose a suitable updating rule, a framework for the AS that can also be applied to its variants like MMAS was introduced in Blum *et al.* [13] and Blum and Dorigo [14]. The main idea is to normalize the bounds of the pheromone trails in the range [0,1], and consequently normalize the quality function of a solution, $f(\cdot)$. The updating rule thus becomes:

$$\tau_i = \tau_i + \rho(\Delta\tau_i - \tau_i). \qquad (12)$$

with

$$\Delta\tau_i = \begin{cases} {}^{1}\!/\!_{f(S_{best})} \Big/ \sum_j {}^{1}\!/\!_{f(Sj)}, & \text{if } i \in S_{best} \\ 0, & \text{otherwise} \end{cases}. \qquad (13)$$

for each node i, where S_{best} is the best solution found and S_j is the solution found by ant j. In this case, τ_i, $\forall i$, is initially set to 0.5 in order to equal the chances in both directions.

4.5 Adaptation to the CPMP

To apply the MMAS to the capacitated *p*-medians problem using the new updating rule, some modifications had to be introduced to take full advantage of all the problem information available. First, τ_{min} and τ_{max} were set to 0.001 and 0.999, respectively, and the pheromone trail was initialized to 0.5.

It can be noticed from Eq. (12) that in order to have a positive increase in the pheromone level it is necessary that $\tau_t < \Delta\tau_t$, and, as the solutions obtained per iteration have a value near the best so far, Eq. (13) will hardly produce a value greater than 0.5, so there is a bound lower than τ_{max} imposed by $\Delta\tau_t$.

For this reason it is proposed a new updating rule, presented in Eq. (14), where a simpler calculation taking into account just the two solutions used to update the pheromone is made, giving a better quality function.

$$\Delta\tau_i = \begin{cases} 1 - \dfrac{l_{best} - g_{best}}{l_{worst} - l_{best}} & i \in \{g_{best}, l_{best}\} \\ 0 & \text{otherwise} \end{cases}. \qquad (14)$$

where g_{best}, l_{best}, and l_{worst} are the global best, local best and local worst solutions, respectively. In Eq. (14), whenever the local best is better than the global one, the pheromone is updated proportionally to the difference between them. Otherwise, the complementary value is taken; thus, the closer the local best from the global best, the closer $\Delta\tau_t$ becomes to one. Note that the global best information is updated after this process, so Eq. (14) can result in a range [0,2], meaning that whenever a new best solution is found, the vertices with low pheromone values are set to a value near or equal to τ_{max} and those which are already with a high value will be equal to τ_{max}.

4.6 Pheromone Stagnation Control

A stagnation control mechanism for the algorithm is proposed so that it is restarted every time it stagnates. For this problem, it is intuitive that when the pheromone trail converges, p points (number of medians) will be at the upper bound, τ_{max}, and the remaining will be at the lower bound, τ_{min}. Thus, every time the sum of all pheromone follows Eq. (15), the algorithm is said to have stagnated and is thus restarted:

$$\sum_i \tau_i = p \cdot \tau_{max} + (n - p) \cdot \tau_{min}. \qquad (15)$$

5 Performance Evaluation

To evaluate the performance of the modified algorithm, several CPMP instances from the literature were tested. For each instance of a given set, it was made the calculation of the *relative percentage deviation* from the best known solution: $RPD = 100 \times (S_{MMAS} - S_{best})/S_{best}$ and then the average was taken, where S_{MMAS} is the best solution found by MMAS and S_{best} the best solution known for each instance.

The first experiment was performed to assess the influence of the information heuristics on the MMAS. Simple experiments were run on the classic instances of Osman [1] and Lorena [15]. Table 1 presents some results found with two sets of parameters α and β to illustrate the importance of the heuristic information. 500 iterations of the improved MMAS (IMMAS) were run with the following parameters: $\alpha = 1$, $\beta = 0$ (only pheromone and no heuristic information), and $\alpha = 3$, $\beta = 1$ (with heuristics but privileging pheromone). As can be seen from Table 1, the heuristic information successfully improves the results found by the Ant System (i.e. only using pheromone information).

Table 1: Average relative percentage deviation from the best known solution to the Osman and Lorena sets. Influence of the information heuristics. Negative results mean that a solution better than the best solution found so far was found

		$\alpha = 1, \beta = 0$	$\alpha = 3, \beta = 1$
Osman	Average (%)	0.081203	**0.064181**
Lorena	Average (%)	0.090277	**–0.11838**

To illustrate the performance of the two different pheromone updating rules studied, some experiments were performed with the IMMAS algorithm applied to the same instance sets as above for 500 iterations. As can be seen from Tables 2 and 3, on harder instances (i.e. larger number of clients and medians to search and harder GAP instances generated from each *p*-median solution) Eq. (14) gives better results than Eq. (13). On easier ones, they both give the same results.

Table 2: Comparison between the two pheromone updating rules in 500 iterations on the first instance set. The first group of columns corresponds to the first 10 instances of Osman [1] and the second group represents the last 10 instances. The best results are presented in bold. "$\Delta\tau$" represents the solution obtained using Eq. (13) while "new $\Delta\tau$" represents the solution obtained using Eq. (14). "*n*" represents the number of clients and "*p*" represents the number of medians for each instance.

	$\Delta\tau$	new $\Delta\tau$		$\Delta\tau$	new $\Delta\tau$
Osman	**Sol.**	**Sol.**		**Sol.**	**Sol.**
	713	**713**		1008	**1007**
	740	**740**		**966**	**966**
	751	**751**		**1026**	**1026**
	651	**651**		**983**	**983**
***n*=50 *p*=5**	**664**	**664**	***n*=100 *p*=10**	**1091**	**1091**
	778	**778**		**955**	**955**
	787	**787**		**1034**	**1034**
	820	**820**		**1043**	**1043**
	715	**715**		**1032**	**1032**
	831	**831**		1007	**1005**

Table 3: Comparison between the two pheromone updating rules in 500 iterations on the second instance set. The best results are presented in bold.

	$\Delta\tau$	new $\Delta\tau$		$\Delta\tau$	new $\Delta\tau$
Lorena	**Sol.**	**Sol.**		**Sol.**	**Sol.**
***n*=100 *p*=10**	17377	**17352**	***n*=300 *p*=30**	41228	**40790**
***n*=200 *p*=15**	**33254**	**33254**	***n*=402 *p*=30**	63966	**62400**
***n*=300 *p*=25**	**45279**	**45279**	***n*=402 *p*=40**	53909	**52857**

In the next experiments, comparisons were performed between MMAS and IMMAS. For each algorithm, 10 trials of 2,000 iterations were run on an Athlon XP + 2000, 1.67GHz, 512 MB RAM running Slackware 9.l, compiled with gcc 3.2, not optimized at compilation.

Table 4 presents the set of problems first introduced in Osman and Christofides [1] and broadly studied in the CPMP literature. The data sets are available at the OR-Library (http://www.brunel.ac.uk/depts/ma/research/jeb/info.html), a repository of test data sets for a variety of Operations Research (OR) problems.

The results were compared with those presented in Osman and Christofides [1], referred to as HSS.OC, which is an implementation of a hybrid involving Simulated Annealing and Tabu Search. As can be observed from this table, the IMMAS algorithm performed better than the MMAS alone and is competitive when compared to the HSS.OC algorithm. It must also be noticed that the variance of the solutions found was 0%, meaning that the same results were found for all 10 trials, indicating the robustness of the algorithm.

Table 5 shows the set of problems created by Lorena in [15], where geographical information about a large city in Brazil was obtained, thus creating a more realistic and complex scenario. In this particular problem, the IMMAS presents a superior performance when compared with the simple MMAS algorithm. Furthermore, IMMAS was capable of finding better solutions than the best solutions known to date. It is also important to observe that the most noticeable differences in performance are on the larger instances, thus suggesting that the proposed modifications help to overcome one important difficulty of Ant Systems, associated with problems containing a large dataset.

6 Discussion and Future Trends

This paper presented the application and further improvements of an ant-colony optimization algorithm to the capacitated *p*-medians problems (CPMP). In particular, it described one form of applying the Max Min Ant

Table 4: MMAS, IMMAS and HSS.OC results for Osman's set of instances, the "Best" column is the best known solution found so far, "Sol." is the solution obtained by each algorithm, "%" is the average relative percentage deviation from best and "Time" is the execution time in seconds, results in boldface are the best found by comparing the algorithms.

		MMAS			IMMAS			HSS.OC	
Osman	**Best**	**Sol.**	**%**	**Time(s)**	**Sol.**	**%**	**Time(s)**	**Sol.**	**%**
***n*=50 *p*=5**	713	**713**	0.00	31.93	**713**	0.00	28.22	**713**	0.00
	740	**740**	0.00	33.52	**740**	0.00	30.11	**740**	0.00
	751	**751**	0.00	45.59	**751**	0.00	37.83	**751**	0.00
	651	**651**	0.00	47.51	**651**	0.00	32.73	**651**	0.00
	664	**664**	0.00	40.24	**664**	0.00	31.98	**664**	0.00
	778	**778**	0.00	39.88	**778**	0.00	33.37	**778**	0.00
	787	**787**	0.00	44.00	**787**	0.00	34.19	**787**	0.00
	820	822	0.24	56.31	**820**	0.00	36.95	**820**	0.00
	715	**715**	0.00	44.05	**715**	0.00	33.64	**715**	0.00
	829	831	0.24	49.12	**829**	0.00	40.12	**829**	0.00
***n*=100 *p*=10**	1006	1008	0.20	316.03	1007	0.09	158.34	**1006**	0.00
	966	**966**	0.00	180.66	**966**	0.00	156.33	**966**	0.00
	1026	**1026**	0.00	180.56	**1026**	0.00	168.29	**1026**	0.00
	982	985	0.30	152.64	**982**	0.00	194.13	985	0.31
	1091	1092	0.09	118.62	**1091**	0.00	154.32	**1091**	0.00
	954	955	0.10	120.84	955	0.10	186.21	**954**	0.00
	1034	**1034**	0.00	150.60	**1034**	0.00	162.23	1039	0.48
	1043	**1043**	0.00	142.65	**1043**	0.00	167.21	1045	0.19
	1031	1033	0.19	118.23	1032	0.09	164.43	**1031**	0.00
	1005	1009	0.40	178.15	**1005**	0.00	214.39	**1005**	0.00
Avg.			0.088			**0.014**			0.049

Table 5: MMAS and IMMAS for Lorena set of instances, the "Best" column is the best known solution found so far, results in bold are the best found among the algorithms, and "Time" is the execution time in seconds, for IMMAS the column "Sol." holds the average result in 10 trials and standard deviation in parentheses, and the "Best" column represents the best solution found along these trials.

		MMAS			IMMAS			
Lorena	**Best**	**Sol.**	**%**	**Time**	**Sol.**	**Best**	**%**	**Time**
***n*=100** / ***p*=10**	17288	17288	0.00	295.95	17264.6 (17.08)	17252	**–0.21**	253.77
***n*=200** / ***p*=15**	33395	33254	–0.42	540.96	33203.7 (12.98)	33187	**–0.62**	1428.10
***n*=300** / ***p*=25**	45364	45251	–0.25	8109.64	45279.1 (32.15)	45245	**–0.26**	1742.26
***n*=300** / ***p*=30**	40635	40638	0.01	7818.13	40662.5 (116.54)	40521	**–0.28**	2127.22
***n*=402** / ***p*=30**	62000	62423	0.68	12701.24	62280,75 (187,24)	62020	**0.03**	1407.44
***n*=402** / ***p*=40**	52641	52649	0.02	10500.15	52627.5 (108.82)	52492	**–0.28**	1484.44
Avg.			0.006				**–0.271**	

System (MMAS) to the CPMP problem that includes a local search heuristics, and combines the MMAS with a new updating rule and a recent framework from the literature in order to improve the performance of the algorithm, mainly when large instances are considered. It is an extension of a previous work by the authors [5].

With the extensions proposed here, based on the framework presented in Blum *et al.* [13] and Blum and Dorigo [14], the results obtained showed that the modified algorithm is competitive and sometimes better than other heuristics found in the literature when applied to the same problem instances. It could also be noted that, over ten trials, the variance in the behavior of the modified algorithm was very small, sometimes zero, for the smaller instances.

Despite the quality of the results already achieved, there are some important aspects that deserve further investigation. For instance, even though the density function gives information about promising medians, it is only accurate on the choice of the first points, because, as a point is chosen, the density surface of the search space changes accordingly. So, recalculating this value for all the candidates not yet chosen could improve the quality of the results obtained, but with the drawback of a high computational cost. Furthermore, it must also be investigated an adaptive distribution of the importance factors given for the pheromone and information heuristic (α and β) so as to try to improve the performance of the algorithm. This happens because, initially, the algorithm has no information about the pheromone trail. Thus, a higher importance should be given to η during the first iterations, and after a number of iteration steps the pheromone trail can give better information than η, so it must have a higher importance as well. Finally, a better GAP local search, or even a constructive heuristics, can be implemented to further improve the assignment of clients to medians.

Acknowledgements

The authors thank CAPES, FAPESP and CNPq for the financial support, and the reviewers for their relevant comments and suggestions.

References

[1] OSMAN, I. H.; CHRISTOFIDES, N., Capacitated Clustering Problems by Hybrid Simulated Annealing and Tabu Search. Pergamon Press, England, 1994, Int. Trans. Opl. Res. v. 1, n. 3, pp. 317-336, 1994.

[2] GAREY, M.R. & JOHNSON, D. S., Computers and Intractability: A Guide to the Theory of NP-Completeness. San Francisco: Freeman. 1979.

[3] DORIGO M. Optimization, Learning and Natural Algorithms. Ph.D.Thesis, Politecnico di Milano, Italy, in Italian, 1992.

[4] DORIGO, M. & DI CARO, G. The Ant Colony Optimization Meta-Heuristic. In D. Corne, M. Dorigo and F. Glover (Eds.), *New Ideas in Optimization*. McGraw Hill, London, UK, Chapter 2, pp. 11-32. 1999.

[5] DE FRANÇA, F. O., VON ZUBEN, F. J., DE CASTRO, L. N. Definition of Capacited p-Medians by a Modified Max Min Ant System with Local Search In: ICONIP - 2004 11th International Conference on Neural Information Processing - SPECIAL SESSION ON ANT COLONY AND MULTI-AGENT SYSTEMS, 2004, Calcutta. Lecture Notes in Computer Science. v. 3316. pp. 1094 – 1100, 2004.

[6] DORIGO, M. & STÜTZLE, T. The Ant Colony Optimization Metaheuristic: Algorithms, Applications, and Advances. In F. Glover and G. A. Ko-

chenberger, *Handbook of Metaheuristics*, Kluwer Academic Press, Chapter 9, pp. 251-286. 2003.

[7] DE CASTRO, L. N. & VON ZUBEN, F. J. (eds.) *Recent Developments in Biologically Inspired Computing*, Idea Group Inc. 2004.

[8] STÜTZLE, T. & HOOS, H.H. The MAX-MIN ant system and local search for the traveling salesman problem. In T. Bäck, Z. Michalewicz, and X. Yao, editors, Proceedings of the IEEE International Conference on Evolutionary Computation (ICEC'97), IEEE Press, Piscataway, NJ, USA, pp. 309-314, 1997.

[9] STÜTZLE T. & DORIGO M. ACO algorithms for the Quadratic Assignment Problem. In D. Corne, M. Dorigo, and F. Glover, editors, New Ideas in Optimization, pp. 33-50. McGraw-Hill, 1999.

[10] AHMADI, S. & OSMAN, I.H., Density based problem space search for the capacitated clustering problem. Annals for Operational Research, 2004 (in press).

[11] RESENDE, G.C.M. & WERNECK, F. R. On the implementation of a swap-based local search procedure for the p-median problem. Proceedings of the Fifth Workshop on Algorithm Engineering and Experiments (ALENEX'03), Richard E. Ladner (Ed.), SIAM, Philadelphia, pp. 119-127, 2003.

[12] TEITZ, M. B.; BART, P., Heuristic Methods for Estimating the Generalized Vertex Median of a Weighted Graph. Operation Research, 16(5):955-961, 1968.

[13] BLUM, C., ROLI, A. & DORIGO, M. HC-ACO: The hyper-cube framework for Ant Colony Optimization. In Proceedings of MIC'2001 - Metaheuristics International Conference, v. 2, Porto, Portugal, pp. 399-403, 2001.

[14] BLUM, C. & DORIGO, M., The Hyper-Cube Framework for Ant Colony Optimization. IEEE Transactions on Systems, Man and Cybernetics, Part B, 34(2): 1161-1172, 2004.

[15] LORENA, L.A.N. & SENNE, E.L.F. Local Search Heuristics for Capacitated P-Median Problems, Networks and Spatial Economics 3, pp. 407-419, 2003.

Application of Ant-based Template Matching for Web Documents Categorization

Siok Lan Ong , Weng Kin Lai, Tracy S. Y. Tai and Kok Meng Hoe
MIMOS, Technology Park Malaysia,
57000 Kuala Lumpur, Malaysia.

Choo Hau Ooi
University of Malaya,
50603 Kuala Lumpur, Malaysia

Keywords: Swarm intelligence, Ant colony optimization, Data clustering, Document clustering

Received: February 18, 2005

The self-organization behavior exhibited by ants may be modeled to solve real world clustering problems. The general idea of artificial ants walking around in search space to pick up, or drop an item based upon some probability measure has been examined to cluster a large number of World Wide Web (WWW) documents. However, this idea is extended with the direct application of template matching with a Gaussian Probability Surface (GPS) to constrain the formation of the clusters in pre-defined areas of workspace with these multi-agents in this paper. Some comparisons between the clustering performance of supervised ants using GPS against the typical ants clustering algorithm are shown. Their performance are evaluated on the same dataset consisting of a collection of multi-class web documents. Finally, the paper concludes with some recommendations for further investigation.

Povzetek: Tehnike kolonij mravelj so bile uporabljene za kategorizacijo internetnih dokumentov.

1 Introduction

Social insects make up 2% of all species of living organisms that live in this world [2], with ants forming by far the largest group - 50% of these social insects are ants. Within the ant colonies, there is specialization in the tasks that need to be performed. Many of these simple but yet important tasks are very similar to some of the real world problems for humans. For example, the foraging behavior of ants has shown to be a useful computing paradigm for solving discrete optimization problems [3]. Similarly, the self-organizing behavior of ants may be used to model intelligent applications such as clustering. This paper will focus on the task performed by the specialized worker ants that include nest and cemetery maintenance through clustering, and model it to cluster the fast growing sources of online text documents in particular.

Similar to any typical document clustering task, web documents clustering may generally be seen as dividing the set of documents into homogeneous groups with the main purpose that documents within each cluster should be similar to one another while those which are from different clusters should be dissimilar [4]. Even though this sounds simple enough, unfortunately, the sheer size of the World Wide Web makes it difficult to manually categorize the documents. In order to automate the process, different well-established clustering approaches have been widely applied to effectively organize the documents based on the above principle in terms of processing time, quality of clustering and spatial distribution. The straightforward model which ants move randomly in space to pick up and deposit items on the basis of local information has also been explored to cluster such web documents [1]. However, it has also been observed that some species of ants combine these self-organisation activities with template mechanisms [5]. A template is a kind of *tool* used by the insects to guide them perform their activities better. For example, in the context of nest building, the shape of the nest may be predefined by templates. The insects will then just build their nest along the markers on such a blueprint.

This paper examines the direct implementation of a template based on a *Gaussian Probability Surface* (GPS) to supervise these homogeneous multi-agents to form clusters within a specified dropping zone. In addition, the results will also be compared with those obtained through unsupervised multi-agents clustering. Basically, the main idea is building on the concept of "*self-organisation along a template*", whereby a template mechanism is combined with the self-organisation mechanism. More specifically, it involves mapping each pixel in the workspace layer to a similar pixel in another surface within the same relative spatial location. Combining the underlying self-organizing mechanisms of the algorithm with templates allows all the items be deposited in some particular regions of space [6].

This paper is organized as follows. In section 2, the key issues of document representation are introduced. This is followed by a description of some theoretical aspects of homogeneous multi-agents in ant colonies in

section 3. The supervised form of this computing paradigm, involving GPS is explained in greater details in section 4. The experimental set-up and the results obtained are shown in sections 5 and 6 respectively. Finally, some conclusions as well as areas for further investigation are discussed in section 7.

2 Web Document Representation

The clustering process of documents, in this case, web pages, involves implementing suitable clustering techniques to group together documents that have similar characteristics. However, before the similar documents can be grouped together, an important process is to identify and extract all the relevant features of each document, so that each document is now represented in a form that the clustering algorithm can process. The feature extraction of a document basically involves finding the representation of the word vector or set of descriptors that best describe it. Concise representations are usually derived from the contents of more complex object. In the case of textual objects i.e. documents (more specifically, web pages), words taken directly from the document but augmented with weights to form a *bag-of-words* representation while disregarding the linguistic context variation at the morphological, syntactical, and semantically levels of natural language[8].

2.1 Automated Text Processing

Automated text processing is the process of producing document representations or "*bags of words*" (also known as index terms) automatically. Conventionally, text processing which follows a standard procedure, may be divided into 4 major text operations [8]. This will be described further in the next few sections.

2.1.1 Lexical Analysis

This is generally defined as the process of converting a stream of characters (the text of the documents) into a stream of words (the candidate words to be adopted as index terms and it involves more than "*linear analysis*" or "*scanning*" of spaces between the words as word separators. The stream of characters making up the text is read one at a time and grouped into *lexemes (l*exemes are minimal lexical unit of a text). However, there are four particular cases that need to be considered with care, viz.

- Digits
- Hypens
- Punctuation marks
- Letters

Many of these characters, especially hypens, digits, and punctuation marks are removed from any further consideration, as shown in figure 1(a) and (b). Once the text have been processed, these lexemes will be fed into another stage for further processing, in this case, to eliminate the stopwords.

The Requests for Comments (RFC) document series is a set of technical and organizational notes about the Internet (originally the ARPANET), beginning in 1969. Memos in the RFC series discuss many aspects of computer networking, including protocols, procedures, programs, and concepts, as well as meeting notes, opinions, and sometimes humor. For more information on the history of the RFC series, see "30 years of RFCs".

(a) The original text

The Requests for Comments RFC document series is a set of technical and organizational notes about the Internet originally the ARPANET beginning in Memos in the RFC series discuss many aspects of computer networking including protocols procedures programs and concepts as well as meeting notes opinions and sometimes humor For more information on the history of the RFC series see years of RFCs

(b) Lexemes

Figure 1: Lexical Analysis

2.1.2 Stopwords Elimination

Stopwords are very commonly used words, and in the English language these would be articles, pronouns, adjectives, adverbs and prepositions which have been known to make poor index terms. They are usually removed from further consideration as index terms when identified in a document. The process of stopwords elimination is illustrated in figure 2 with a part of Dr. *Martin Luther King Jr.'s* well-known "*I Have a Dream"* speech that he delivered in Washington D.C. on August 28, 1963.

I have a dream that one day this nation will rise up and live out the true meaning of its creed: We hold these truths to be self-evident: that all men are created equal. I have a dream that one day on the red hills of Georgia the sons of former slaves and the sons of former slave owners will be able to sit down together at a table of brotherhood. I have a dream that one day even the state of Mississippi, a desert state, sweltering with the heat of injustice and oppression, will be transformed into an oasis of freedom and justice. I have a dream that my four children will one day live in a nation where they will not be judged by the color of their skin but by the content of their character. I have a dream today.

Total Word Count : *143*

(a) *The original text*

dream day nation rise live true meaning creed: hold truths self-evident: created equal. dream day red hills Georgia sons former slaves sons former slave owners able sit table brotherhood. dream day Mississippi, desert state, sweltering heat injustice oppression, transformed oasis freedom justice. dream children day live nation judged color skin content character. dream today

Total Word Count : ***54***

(b) *With stopwords removed*

Figure 2: Process of Stopwords Elimination

Notice that there is now a significant reduction in the number of words amounting to about 62.2% of the total initial amount after the stopwords elimination process.

Closer examination at the first sentence of the address shown here will clearly reveal that common words like *I, have, a, that, one, this, will, up, and, out, the, of,* and *its*, have been eliminated as stopwords.

2.1.3 Stemming

The objective of stemming [8] is to remove affixes (i.e. *prefixes* and *suffixes*) so as to reduce the total size of the index terms. This is normally done with ways of finding morphological variants of terms in documents. Examples of the stemmed index terms are shown in Table 1 below.

Table 1: Example of the stemming process on several terms

Original Term	**After stemming**
possibilities	*possibl*
Possible	*possibl*
possibility	*possibl*
Possibly	*possibli*
Software	*Softwar*
Software	*softwar*

2.1.4 Indexing

This is the final process of text processing where the index terms are extracted to identify the important features for each document. Feature extraction of a document involves finding the optimal representation of the word vector or set of descriptors that best describe the salient features of the documents.

There are several ways to shortlist the index terms from a document. In this paper, the index terms were selected by using one of the most commonly used weighting approach known as *tfidf* (term frequency inverse document frequency):

$$w_{ij} = f_{ij} \times \log\left(\frac{N}{n_i}\right) \quad (1)$$

where w_{ij} is the weight of word i in document j, f_{ij} be the frequency of word i *in* the document j, N the number of documents in the collection, and n_i the total number of times word i occurs in the whole collection.

3 Ant Colony Models

The ability of insects such as ants living in a colony has fascinated many in the scientific community and this has led to more detailed studies on the collective behavior of these creatures. Even though these insects may be small in size and live by simple rules, but yet they are able to survive well within their colony. Scientists have recently found that this behavior could be borrowed to solve complex tasks such as text mining, networking etc. *Deneubourg* et al. had developed this concept further by modeling the ant's action in organizing their nests for data classification. Assuming each of these multi-agents carries one item at a time and there is only one item type, the probabilistic functions, P_p and P_d that model such behaviour are shown below, i.e.:

$$\text{Picking up probability, } P_p = \left(\frac{k_1}{k_1 + f}\right)^2 \quad (2)$$

$$\text{Dropping probability, } P_d = \left(\frac{f}{k_2 + f}\right)^2 \quad (3)$$

where f denotes the fraction of similar items in the neighbourhood of the agent, while k_1 and k_2 are threshold constants. When $f \to 1$, it means that there are many similar items in the neighbourhood, indicating that there is a high possibility that the multi-agent will put down the item it is carrying, as P_d will be high too. Similarly, the agent is not likely to pick up the item when P_p is low. This will happen when most of the items in the neighbourhood are dissimilar, as indicated by $f \to 0$. Essentially, there is a high possibility of picking up items which are isolated and transporting them to another region where there are now more of its kind in the neighbourhood. The possibility of dropping the item will be low when $P_d \to 0$.

Lumer & Faieta (LF) [9] had reformulated *Deneubourg* et al.'s [10] model to include a distance function, d between data objects for the purpose of exploratory data analysis. The binary distance between objects i and j, $d(o_i, o_j)$, is assigned 1 for dissimilar objects and 0 for similar objects. Essentially this binary distance measure is the Hamming distance [11] between two objects. The fraction of items in the neighbourhood, f in equation (2) and (3) is replaced with the local density function, $f(o_i)$ which measures the average similarity of object i with all the other objects j in its neighbourhood, N. Given a constant, α, and the cells in a neighbourhood $N(c)$, $f(o_i)$ may be defined as:

$$f(o_i) = \frac{1}{|N^2(c)|} \sum_{o_j \in N(c)} \left[1 - \frac{d(o_i, o_j)}{\alpha}\right] \text{ if } f > 0$$

$$\text{Otherwise, } f(o_i) = 0 \quad (4)$$

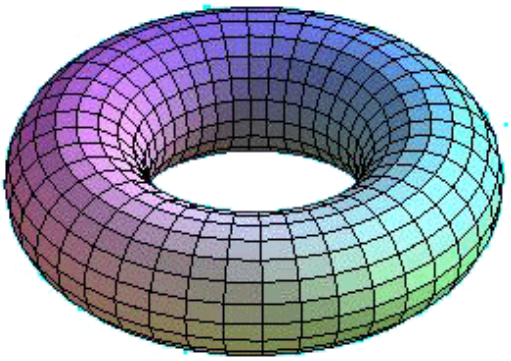
α is a factor that defines the scale of dissimilarity that will influence when two items should or should not be placed together in the same neighbourhood. For exxample, if α is large, it will only marginalise the differences between the items, leading to the formation of clusters composed enitrely of items which should not be grouped together in the same cluster.

4 Supervised Ant Colony Models with *Gaussian Probability Surface* (GPS)

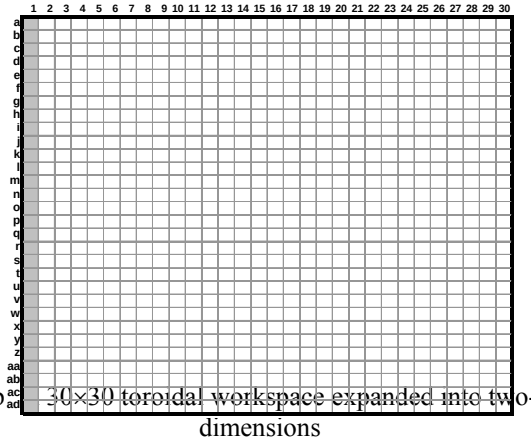
In several species of ants, the worker ants are known to perform corpse aggregation and brood sorting where the clusters formed is at arbitrary locations [12]. However,

there are other species, like the *Acantholepsis custodiens* ants that are known to perform self-organization which are constrained by templates [5]. A template is a pattern that is used to construct another pattern. In the case of some species of the ants which are found in Nature, they utilize the information related to the temperature and humidity gradients in their surroundings to build their nests to spatially distribute their brood [5]. This concept of self-organizing with templates has been used by *Dorigo & Theraulaz* for data analysis and graph partitioning [6].

With such mechanisms, the result is that the final structures would closely follow the configuration defined by the templates. However, this is only useful in applications where the numbers of clusters are known beforehand. The template we have used here, in the form of a *Gaussian Probability Surface* (GPS) guides the multi-agents to form clusters within a *toroidal* working space.



(a) Example of a toroidal surface



(b) 30×30 toroidal workspace expanded into two-dimensions

Figure 3: Toroidal Workspace

Hence, all the cells that are lying at the perimeter of this workspace will be adjacent to each other. For example, the cells in column ***1*** of a 30 × 30 workspace will have neighbours in columns ***2*** and ***30***, as shown in figure 3 above. Similarly, the cells in the top **row a**, will have **row b** and ***row ad*** as their neighbours.

The GPS equation, P(x,y) is shown in equation 5 below.

$$P(x,y) = P_{max} \sum_{i=1}^{n} \left[e^{-\left(\frac{(x-x_{0i})^2+(y-y_{0i})^2}{\sigma^2} \right)} \right] + \delta \qquad (5)$$

where, $0 \le P(x,y) \le 1$,and,

P_{max} maximum value of probability, $0 \le P_{max} \le 1$

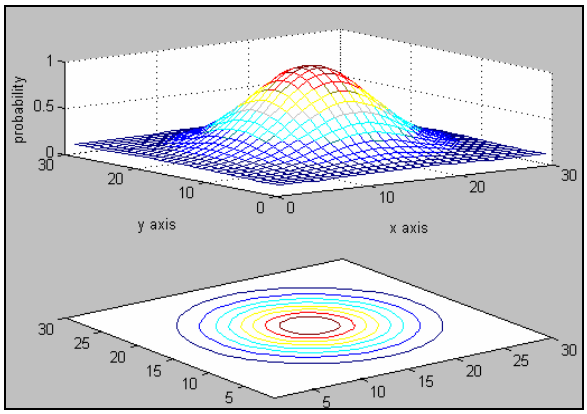
δ offset value.

σ^2 constant defined by user that also determines the steepness of the Gaussian probability surfaces.

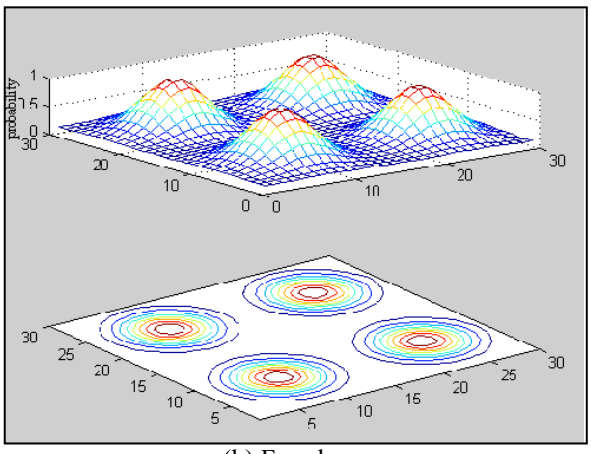
x_{0i}, y_{0i} Coordinates of the centre of each dropping zone(i.e. the peak of the humps).

x, y Coordinates on any single point in the workspace.

i number of humps, $1 \le n \le 5$.



(a) One hump



(b) Four humps

Key :

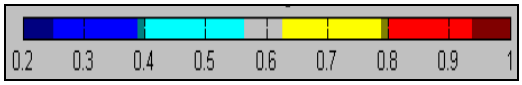
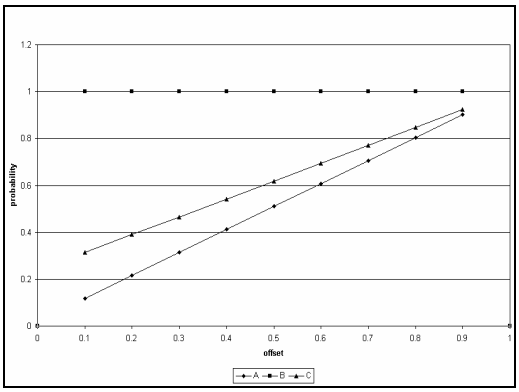


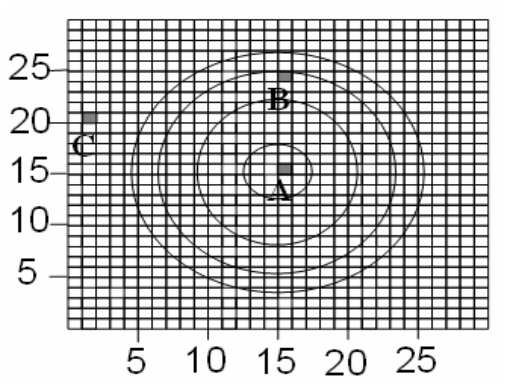
Figure 4: *The Gaussian Probability Surface (GPS) superimposed onto the toroidal working space. The* colour bar indicates the probability of the surface for various colours.

The probability surface is two dimensional and isotropic (circular symmetry). Figure 4 shows an example of the

GPS model with one and four humps superimposed onto the toroidal workspace. Both the x- and y- axes represent the location within this toroidal workspace. As the height of the probability surface increases, the probability of dropping the document by the multi-agents is higher**.** Hence, more similar documents are expected to be clustered in the area underneath the probability "*shadow*" as defined by each hump. Essentially, this will enhance the quality of clustering by having clusters with similar document types in the specified dropping zone instead of forming in non-deterministic region of the workspace.



(a) The plot of the probability values for each of the 3 points monitored.



(b) The location of the 3 locations monitored.

Figure 5: The probability values of 3 locations for various offsets.

Figure 5(a) shows the probability values with different values of the offset δ. It may be seen that the probability for depositing an item increases linearly with an increase in the offset (δ) values. The probability is close to 1 at the peak of the humps and there was only a slight increase for any increments of the offset δ. However, there is a significant change in the probability for points at the lower portion of the surface for different values of the offset. In addition, the dropping probability distribution in the regions between the contour lines does not vary much for higher offsets. This implies that there are actually more space for the multi-agents to unload the documents for higher offsets.

The multi-agents can only move one step in any direction at each time unit from its existing location to an *unoccupied* adjacent cell. Only a single agent and/or a single item is allowed to occupy any one cell at a time. An agent occupying any cell, c on the clustering space immediately perceives a neighbourhood of 8 adjacent cells i.e. $N(c) = 8$. This is illustrated in figure 6 below.

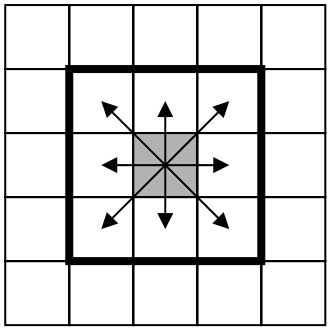


Figure 6: The neighbourhood of any one location in the workspace.

The decision of an *unladen* agent to either pick up or ignore an item o_i at cell c is dictated by a probability P_p that is based on a local density function, $g(o_i)$. This local density function determines the similarity between o_i and other items o_j, where $j \in N(c)$. If an agent *laden* with item o_i lands on an empty cell c, it will calculate a probability P_d based on the same function $g(o_i)$ and decides whether to drop o_i or keep on carrying it. Unlike f (see Eq. 4) which uses a distance measure and an additional parameter α, the function $g(o_i)$ uses a similarity measure which may be defined as follows:

$$g(o_i) = \frac{1}{N(c)} \sum_{o_j} S(o_i, o_j) \qquad (6)$$

where $S(o_i,o_j)$ is a measure of the similarity between objects o_i and o_j.

To model the inherent similarity within documents, one measure that is often employed in information retrieval is the *cosine* measure, where,

$$S_{\cos}(doc_i, doc_j) = \frac{\sum_{k=1}^{r} (f_{i,k} \times f_{j,k})}{\sqrt{\sum_{k=1}^{n} (f_{i,k})^2} \times \sqrt{\sum_{k=1}^{m} (f_{j,k})^2}} \qquad (7)$$

r is the number of common terms in doc_i and doc_j, n and m represent the total number of terms in doc_i and doc_j respectively. $f_{a,b}$ is the frequency of term b in doc_a. A useful property of the cosine measure S_{cos} is that it is invariant to large skews in the weights of document vectors, but sensitive to common concepts within documents.

To guide the multi-agents to drop the documents onto a specified dropping zone within the two dimensional workspace, the concept of a Gaussian Probability Distribution surface overlaid on to this work

space was used. This model requires large samples and repeated measurements with random errors distributed according to the Gaussian probability[13].

5 Experimental Set-Up

The 80 web pages used in the experiment came from four different categories—Business, Computer, Health and Science that were randomly retrieved from the *Google* web directory. These were then pre-processed to extract a representative set of features. The main purpose of this feature extraction process is to identify a set of most descriptive words of a document. This resulted in a collection of 17,776 distinct words. To reduce memory requirements during clustering, the collection was represented by a sparse matrix with three elements per row: (i) a unique web page identifier, (ii) a unique word identifier, and (iii) the frequency value of each word within the web page. This is illustrated in Figure 7 below. Hence, only the most descriptive words will be represented here, with each word assigned a unique ID and its associated frequency within the page.

<**page ID**>, <**wordID**>, <**wordFreq**>
<**page ID**>, <**wordID**>, <**wordFreq**>
<**page ID**>, <**wordID**>, <**wordFreq**>
<**page ID**>, <**wordID**>, <**wordFreq**>
…

(a) : Input representation

357,1,1
463,5,5
179,2379,1
355,2379,3 …

(b): An example

Figure 7 : Input representation of the web documents

One of the inherent challenges of representing these documents with a set of keywords involves the optimal selection of these words for the set of features. Using all the *17,776* words will no doubt leave all the documents represented, but the sheer size of the set of words will incur a very heavy computation overhead. Even then, many of these words do not occur across many of the documents. It is rare to have all the words in the list to occur throughout the whole set of documents. This is illustrated in figure 8 with a sub-set of 2 documents from each of the 4 categories for 20 words. The Y-axis represents the frequencies of the words for each web document. Even though this figure shows only 2 documents from each category it clearly illustrates the occurrences of the words and their frequencies for each document.

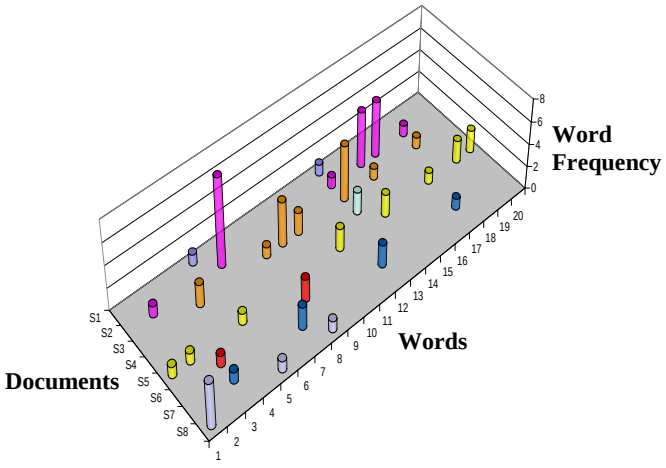


Figure 8: 3D representation of the feature coverage for 8 documents.

Intuitively, it would be good to remove all those features (words) that have very low occurrences across the documents in the set. Good candidates for such a removal would be those that occur in all the documents in any one class of similar documents. In addition, the words which have been extracted for these text documents, were augmented with the appropriate weights, while disregarding the linguistic context variation at the morphological, syntactical, and semantic levels of natural language. The extracted *word-weight* vectors are usually of high dimensions. A total of 6,976 distinctive words were found in this collection of web documents investigated here

Next, the classification of the dataset described in the previous paragraph above using supervised and unsupervised multi-agents within a 30x30 toroidal grid and 15 homogeneous agents with threshold constants, k_1 = 0.01 and k_2 = 0.15 was investigated.

Both the supervised and unsupervised approaches were set to run at a maximum of iterations, t_{max} of 140,000. As there are four document categories in this experiment, a similar number of humps were pre-defined at the start of the supervised approach.

6 Results and Discussion

This section depicts the experimental results of supervised and unsupervised multi-agents clustering with the parameters setting as described in the previous section.

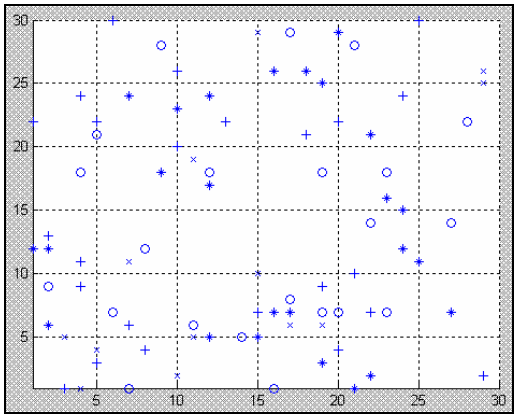


Figure 9(a): The random distribution of the web documents

Key :
o – Business ∇ - Computer + - Health * - Science

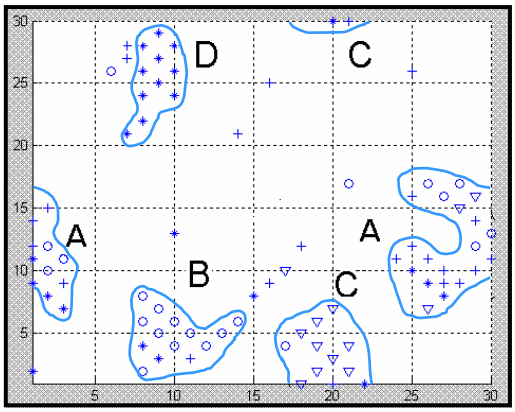


Figure 9(b): Clusters of documents formed at t = *140,000.*

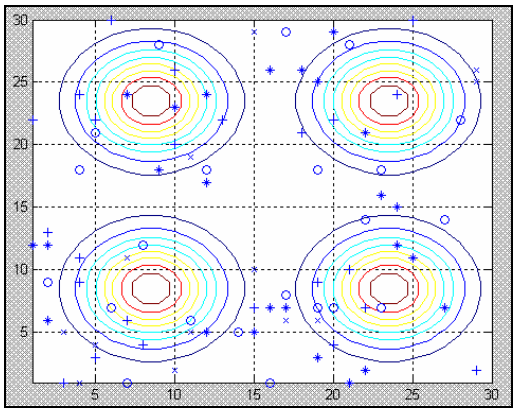
Key :
o – Business ∇ - Computer + - Health * - Science

Figure 9(a) shows how the web documents were initially scattered on the two dimensional workspace at time *t=0*. After 140,000 iterations, four clusters of mixed classes of documents were formed. The quality of the results were evaluated through the measures of purity and entropy. Table 2 below shows the purity (measures the similarity) and entropy (which measures the distribution of various (actual) categories of documents within a cluster) values of the clusters [14]. A high value of purity suggests that the cluster is a *pure* subset of the dominant class. Similarly, an entropy value of 0 means the cluster is comprised entirely of one class. On the other hand, an entropy with a value of 1 would strongly indicate that the cluster itself is a mixture of items without any distinctive or dominant class. The overall entropy value is the weighted sum of the individual values for each cluster which takes into account the size of each cluster formed. The same applies for the overall purity value.

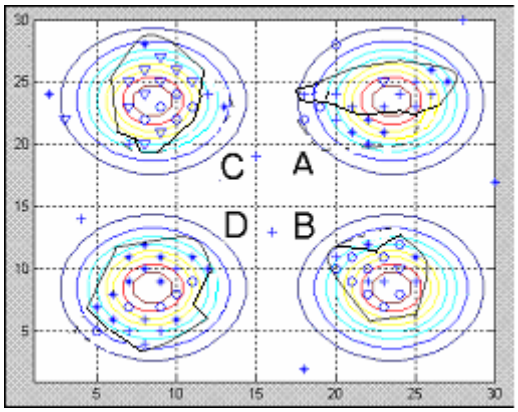
Table 2: The purity and entropy values for different clusters of documents and the overall result.

Cluster	Entropy	Purity	Majority Class
A	0.9297	0.4000	Health
B	0.4732	0.7857	Business
C	0.5337	0.6923	Computer
D	0.0000	1.0000	Science
Overall	**0.5181**	**0.5375**	-

Figure 10 : Graphical representation of the differences in the size of the clusters formed.



(a): The random distribution of the web documents on the workspace at t =*0*.



(b): Four clusters with each containing a majority of different classes were formed at t = *80,000.*

Figure 11: Key : **o** – Business ∇ - Computer + - Health * - Science

Figure 11(a) shows the initial placement of the documents which were scattered on the workspace with the contour plots of the GPS superimposed upon it. After 80,000 iteration it was obvious that the multi-agents had sorted the documents into four different clusters. Most of the documents in the contour regions were closely placed near to the centre of each cluster. In addition, there were only nine documents found scattered at the base of the probability surfaces (indicated by the areas outside the contours). The purity and entropy values obtained from this approach are depicted in Table 3 below:

Table 3: The purity and entropy values for different clusters of documents and the overall result

Cluster	Entropy	Purity	Majority Class
A	0.8229	0.4706	Health
B	0.7960	0.5625	Business
C	0.8390	0.5263	Computer
D	0.7050	0.5715	Science
Overall	**0.3954**	**0.5000**	-

Figure 12 : Graphical representation of the differences in the size of the 4 major clusters formed.

In comparison, although both approaches produce clusters which have nearly similar purity values, however the entropy for the supervised approach using the GPS was approximately 20% better when compared to the unsupervised approach. Moreover, it was also found that it was difficult to identify the clusters if GPS had not been used. In other words, the spatial distribution between clusters was uneven without GPS. Conversely, if GPS was adopted, the clusters in the contour areas could be easily identified as the spaces between the clusters were more distinctive. In addition, the clusters formed by the GPS were neat and more tightly coupled whereas those without GPS were loose as shown in figure 11(b) and 9(b) respectively. A graphical representation of the difference in the size of the cluster formed is shown in figures 10 & 12. These depict the differences in the size of the clusters formed. Both graphical representations were drawn on the same scale. Clearly, there is also a greater uniformity in the size of the clusters generated when the clusters were formed with GPS.

In terms of the stability of the clusters, with GPS, the multi-agents were able to move most documents into the cluster itself and seldom went beyond the specified regions. Any document could be easily moved around the workspace when the agents were fully unsupervised. Hence, it is suggested that the GPS was actually guiding the multi-agents to cluster the documents while constraining the size of clusters in certain regions. More importantly, there was also an improvement in processing time required. With the supervised approach, the clusters were formed at 80,000 iterations, as compared with 140,000 for the unsupervised approach. This would be very useful for the retrieval and access of high dimension web documents.

7 Concluding Remarks and Future Directions

In this paper, the findings of an extended study on using a multi-agent system based on the collective behavior of social insects i.e. ants, to cluster web documents retrieved from a popular search engine were presented. Unlike earlier work, the concept of a direct application of the concept of a Gaussian Probability Surface (GPS) to constraint the formation of the clusters in pre-defined areas in the workspace was introduced. The experimental results showed that the proposed multi-agent system was able to induce clusters with better characteristics than those obtained without this probability surface, even though these results may only be marginally better Visually it may also be obvious that the clusters are better formed than those obtained when there is no GPS. In figure 9(b) one can clearly identify two smaller clusters. This is illustrated in figure 13 below.

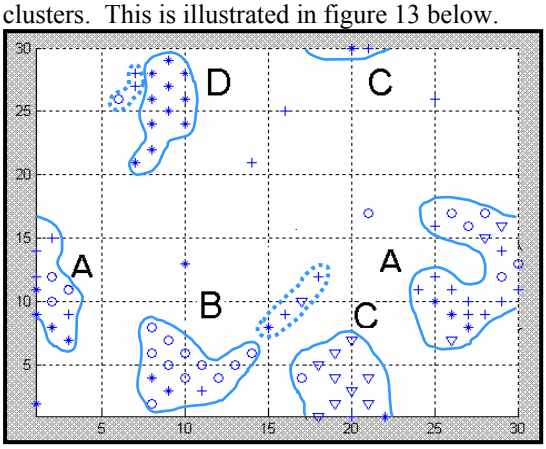


Figure 13: 6 clusters of documents formed with the unsupervised approach

Essentially, unlike the GPS-driven clustering approach, without specifying the exact number of clusters to be formed, the unsupervised approach has formed a total of 4 large clusters and two smaller ones. The results obtained, although not on par with the classification ability of human experts, do demonstrate the potential and effectiveness of ant-like multi-agent systems in handling complex and high-dimension data clusters.

In conclusion, the new approach to organize web documents with the Gaussian probability surface have shown some interesting and improved results. It has also been noticed that the offsets do have a profound effect on the quality of the clusters formed as well as the speed of convergence of the multi-agent system. Obviously, if the offset of the GPS is high, the multi-agents would have a higher freedom to drop the web documents that they may be carrying, over a wider area. A low offset has the opposite effect. Hence, it would be interesting to explore how a non-stationary offset can produce better clusters in the future. Future investigations should also focus on a larger perceivable time-dependent neighbourhood for agents and a better formulation of a stopping criterion based on homogeneity and spatial distribution of clusters. Lastly, future research efforts should also look at developing a deterministic initial distribution of data points on the workspace which may improve the clustering results.

Acknowledgements

The authors would like to thank *C.C. Loy* for his help in generating the contour graphs. *W.K. Lai, Tracy S.Y. Tai and K.M. Hoe* gratefully acknowledges the research grant

provided by the Malaysian *Ministry of Science, Technology & Innovation* under grant *04-01-04-005 EA 001* that have in part resulted in this article.

References

[1] Hoe K. M., Lai W.K., & Tracy Tai, "*Homogeneous Ants for Web Document Similarity Modeling and Categorization*", Proceedings of the Third International Workshop on Ant Algorithms, pp 256 – 261, September 12th – 14th, 2002, Brussels, Belgium.

[2] M.Dorigo, *Artificial Life: "The Swarm Intelligence Approach"*, Tutorial TDI, Congress on Evolutionary Computing, Washington, DC. (1999).

[3] Engelbrecht, A.P., "*Computational Intelligence: An Introduction*", John Wiley & Sons Ltd (2002), ISBN: 0-470-84870-7.

[4] J.Handl, J.Knowles and M.Dorigo, "*Ant Based Clustering: a comparative study of it's relative performance with respect to k-means, average link and ld-som",* http://wwwcip.informatik.uni-erlangen.de/~sijuhand/TR-IRIDIA-2003-24.pdf, March 24th 2004.

[5] Bonabeau, E., Dorigo, M., and Theraulaz, G., "*Swarm Intelligence: From Natural to Artificial Systems",* University Press, Oxford (1999), pp 184.

[6] Bonabeau, E., Dorigo, M., and Theraulaz, G., "*Swarm Intelligence: From Natural to Artificial Systems",* University Press, Oxford (1999), pp 199.

[7] Baeza-Yates, R.. and Ribeiro-Yates, B., "*Modern Information Retrieval",* ACM, NY (1999).

[8] M. F. Porter, "*An Algorithm for Suffix Stripping*", in Program, Vol. 14 (3) (1980), pp. 130-137.

[9] Lumer, E.D. and Faieta, B., "*Diversity and Adaptation in Populations of Clustering Ants",* Int. Conf. Simulation of Adaptive Behavior: Fr. Animals to Animats. MIT, MA (1994).

[10] Deneubourg, J. L., Goss, S., Franks, N.R., Sendova-Franks, A., Detrain, C., and Chretien, L., "*The Dynamics of Collective Sorting: Robot-like Ants and Ant-like Robots",* Int. Conf. Simulation of Adaptive Behaviour: Fr. Animals to Animats. MIT, MA (1990).

[11] Definition of Hamming Distance, National Institute of Standards & Technology, Available at http://www.nist.gov/dads/HTML/hammingdist.html, 25/1/2005.

[12] Bonabeau, E., Dorigo, M., and Theraulaz, G., "*Swarm Intelligence: From Natural to Artificial Systems",* University Press, Oxford (1999), pp 149.

[13] Department of Physic and Astronomy, "Physics and Astronomy: The Gaussian Distribution", http://physics.valpo.edu/courses/p310/ch2.3_gaussian/, March 24th 2004.

[14] Steinbach, M., Karypis, G., and Kumar, V., "A Comparison of Document Clustering Techniques", KDD Workshop on Text Mining (2000).

Efficient and Scalable Communication in Autonomous Networking Using Bio-inspired Mechanisms – An Overview

Falko Dressler
Dept. of Computer Science 7
University of Erlangen-Nuremberg
Martenstr. 3
91058 Erlangen, Germany
dressler@informatik.uni-erlangen.de
http://www7.informatik.uni-erlangen.de/~dressler/

Keywords: Bio-inspired networking, Organic computing, Scalability, Communication, Autonomous networking

Received: October 11, 2004

Autonomous networking is a challenging research area. Systems, in this case networks, are composed of many independent entities that perform a predefined task. The behavior of the global system is a result of the interaction of all the autonomous entities. The programming paradigms shift to the development of just these small entities. Self-organization is the solution for managing such environments. In this paper we demonstrate the possibilities which evolve by the application of cell biology for computer networking. With the focus on autonomous networking, the combination with methodologies known from swarm intelligence is evaluated. We show the capabilities of this combination and derive destinations and goals for self-organization in communication networks showing a more efficient and scalable behavior.

Povzetek: Pregled komunikacij v avtonomnih mrežah na osnovi bioloških mehanizmov.

1 Introduction

Besides to classical research area of bioinformatics, the turn to nature for solutions to technological questions has brought us many unforeseen great concepts. This encouraging course seems to hold on for many aspects in technology. Many efforts were made in the area of computer technology employing mechanisms known from biological systems. The most known examples are swarm intelligence, evolutionary or genetic algorithms, and the artificial immune system. The adapted mechanisms find application in computer networking for example in the areas of network security [7, 15], pervasive computing [11, 27], and sensor networks [21, 23].

Organic computing, which is a new term covering the bio-inspired mechanisms in engineering and computer science related fields, is attempting to build high-scalable architectures, which are self-organizing, self-maintaining, and self-healing [6, 14]. In the fields of computer networking, methods known from swarm intelligence are employed. Mechanisms known from ant colonies and swarms of bees build the basis for cooperative tasks [22, 25]. Issues of self-organization are directly addressable using such mechanisms.

In contrast, the focus of our group lays on trying to map the cellular and molecular biology to networking architectures [11, 12, 20]. Recently, it was shown that the known approaches to study effects in computer networking, especially methods to analyze the behavior of large scale networks suffer from many presumptions. We try to study this behavior by analyzing the internal functioning of network components as well as there interactions in comparison with cellular systems and the associated intra and extra cellular signaling pathways.

The main focus of this work is to show the similarities of computer networks and cellular systems. Based on the knowledge about cellular metabolism, new concepts for the behavior patterns of routers, monitor systems, and firewalls can be deduced and the efficiency of individual sub-systems can be increased. Focusing on examples of hot topics in the computer society, i.e. network security, potential solutions motivated by cellular behavior are currently studied and, hopefully, will soon bring new results in these areas. In combination with efforts known from swarm intelligence, most self-organization issues can be addressed. Cell biology based mechanisms build new communication paradigms and ant colonies are adapted to group formation. Doing this, we must keep in mind that the deeper the parallels between biology and technology, the more important it is to map the corresponding elements correctly. Algorithms known from swarm intelligence are employed for clustering and group formation issues. This is a necessary basis for further bio-inspired communications.

The rest of the paper is organized as follows. In section 2, an overview to the state of the art in bio-inspired networking is provided. This builds the basis for

further studies discussed in section 3. The combination with swarm intelligence is detailed in section 4. New destinations and goals are shown in section 4 and some conclusions summarize the paper.

2 Bio-inspired Networking – State of the Art

Since billions of years, nature worked out organisms that adapted perfectly to environmental changes. Survival of the fittest is the primary selection mechanisms. Research on self-organization started in the 1960ies and combination of nature and self-organizing technical systems was first introduced by Eigen [13]. Reviews of current research on biological self-organization can be found in [4, 14].

The development in the area of bio-inspired engineering is relying basically on the artificial immune system, swarm intelligence, evolutionary (genetic) algorithms, and cell and molecular biology based approaches. The immune system of mammals builds the basis for research on the artificial immune system. The reaction of the immune system, even to unknown attacks, is a high-adaptive process. Therefore, it seems obvious to apply the same mechanisms for self-organization and self-healing operations in computer networks [16, 19].

The behavior of large groups of interacting small insects such as ants and bees builds the basis for field of swarm intelligence. Simple and unrelated autonomously working individuals are considered to compose complex cooperative tasks. Similar actions are required in various areas of engineering and computer science and should build a basis for building self-organizing systems [3, 18]. A main focus lies on the formation of groups or clusters.

Evolutionary (genetic) algorithms are self-manipulating mechanisms. The evolution in nature is the basis for such methodologies. In particular, there are multiple ways for organisms to learn. A natural selection process (survival of the fittest) is going on letting only the optimal prepared organisms to survive and to reproduce. Changes appear for example by mutations. An overview to evolutionary algorithms is provided for example in [2, 6].

An emerging research area looks for cell and molecular biology based approaches. All organisms are built in the same way. They are composed of organs, which consist of tissues and finally of cells. This structure is very similar to computer networks, which is also true for the signaling pathways. Therefore, research on methods in cell and molecular biology promises high potentials for computer networking in general and adaptive sensor networks and network security in particular [10, 11, 12, 20].

Optimization in general and routing aspects are also in the focus of bio-inspired networking. Swarm intelligence builds the basis for this kind of work as shown by Di Cargo and Dorigo [8, 9]. The group of Suda has built a middleware, called the bio-networking platform [28, 30], to investigate in multiple biological inspired technologies. Optimizations and application in various research areas such as pervasive computing [27] and security [26] have been worked out. In the area of sensor networks, first attempts to study the behavior of swarms of insects, typically ants and bees, and to adapt the discoveries to build more efficient sensor networks are in progress. For example, Kadrovach [17] and the group of Osadiciw [23, 24, 25] are working in this area. Liang provided an approach for redundancy allocation [22] applicable in autonomous systems environments. Bio-inspired robots, which use biological mechanisms for flight control and localization, were developed for example by Chahl and Thakoor [5, 29].

All these examples demonstrate that there is already work on bio-inspired mechanisms in the fields of communication networks.

3 Cell Biology as the Key for Computer Networking

In this section we focus on cell biology as a key for computer networking. We first show structural similarities followed by some information about the signaling pathways within single cells and between tissues. Based on obvious similarities, high potentials of this analysis are worked out which will lead to paradigms and algorithms showing a higher efficiency in computer networking.

3.1 Structural Comparison of Organisms and Computer Networks

The sector of cell and molecular biology is analyzing the basic behavior of organisms from a microscopic point of view. All the cells are acting in a predefined manner, controlled by the DNA. In summary it can be said, that cells are the smallest entities in a profoundly self-organizing system comparable to autonomous systems in computer science.

In Fig 1, a structural comparison of organisms and computer networks is shown. It can be seen that both show high similarities. This organization of an organism is a highly regulated process from the single cell up to complex organs of the body. The hierarchy in the organism is very high. Every process, e.g. movement, metabolism, communication, etc. is organized by interactions of several organs. Organs represent an assembly of one or more tissues, which fulfill a common function. One tissue is build by different cell types. One cell type consists of identical cells, which are associated and communicate with each other to fulfill a common function within the tissue.

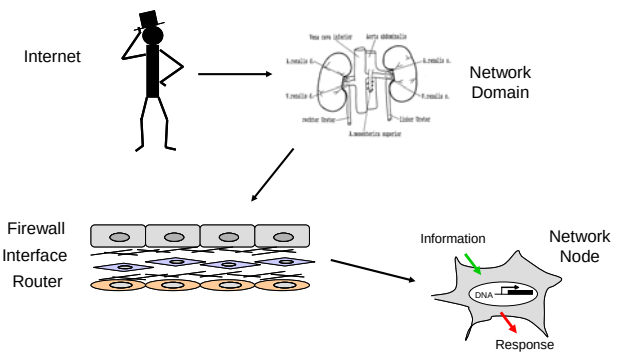


Fig 1. Structural comparison of organisms and computer networks

The organization in computer networks is quite similar. An internet consists of network domains, (sub-)networks, and network nodes, respectively. The network nodes fulfill specific tasks that finally lead to a common behavior of the network. Examples are the storage the aggregation, and the forwarding of data.

3.2 Cellular information exchange and adaptation to network security

The focus of this section is to examine the information exchange in cellular environments and to extract the issues in computer networks that can be addressed by the utilization of these mechanisms [11, 12, 20].

Similar to the structure, the intercommunication within both systems is comparable. Information exchange between cells, called signaling pathways, follows the same requirements as between network nodes. A message is sent to a destination and transferred, possibly using multiple hops, to this target.

From a local point of view, the information transfer works as follows. A specific signal reaches only cells in the neighborhood. The signal induces a signaling cascade in each target cell resulting in a very specific answer which vice versa affects neighboring cells. This process is depicted in Fig 2. A cell is shown with a single receptor that is able to receive a very specific signal and to activate a signaling cascade which finally forms the cellular response.

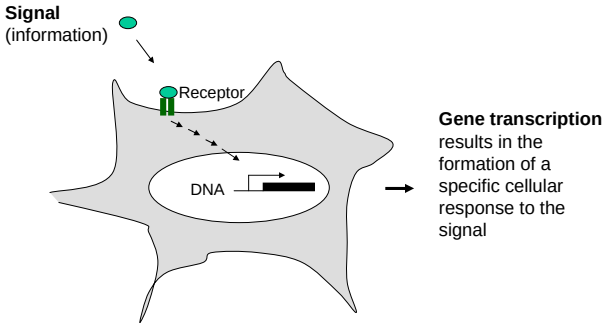


Fig 2. Local information exchange in the cellular environment

Similar mechanisms are present in networking environments. Looking on network security solutions, monitoring probes gather information about the ongoing traffic in the network. The collected data will be sent to an attached intrusion detection system for further processing. Finally, corresponding firewall systems are configured with rules concerning the actual behavior in the network. Similar mechanisms can be deduced from pervasive computing environments or sensor networks. General issues to address in such a network are:

- Adaptive group formation
- Optimized task allocation
- Efficient group communication
- Data aggregation and filtering
- Reliability and redundancy

The remote information exchange works analogue. As depicted in Fig 3, proteins are used as information particles between cells. A signal can be released into the blood stream, a medium which carries it to distant cells and induces an answer in these cells which then passes on the information or can activate helper cells (e.g. the immune system). The interesting property of this transmission is that the information itself addresses the destination. Only cells with a very specific receptor are able to receive the information, i.e. the protein binds at the receptor.

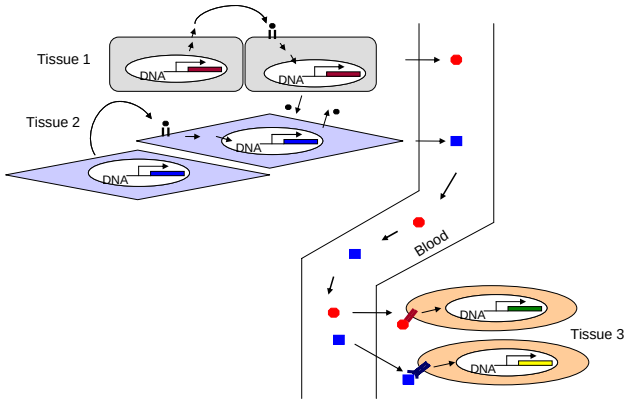


Fig 3. Remote information exchange between cells and tissues

In this scenario, the corresponding issues in computer networking are for example:

- Localization of significant relays, helpers, or cooperation partners
- Semantics of transmitted messages
- Cooperation across domain borders
- Internetworking of different technologies
- Authentication and authorization

The lessons to learn from biology are the efficient and, above all, the very specific response to a problem, the shortening of information pathways, and the possibility of directing each problem to the adequate helper component. Therefore, the adaptation of mechanisms from cell and molecular biology promises to enable a more efficient information exchange. Additionally, issues of task allocation and group communication are directly addressed by the introduced capabilities.

4 Application of Swarm Intelligence

The studies on the behavior of swarms of bees and ant colonies have already shown high potentials solving issues in the areas of group formation and self-organization [9, 22, 23].

Some of the communication aspects, e.g. localization of resources, signaling of control information, and others can be directly addressed by the methodologies known

from cell biology as described above. Others need further assistance. The most challenging issue is the adaptive formation of groups of nodes, so called clusters. Communication mechanisms provided by intercellular signaling pathways typically rely on the formation of clusters.

Groups can be organized using various parameters such as similar properties, localization information, and other relationships. Swarm intelligence and collective behavior are key solutions for such questions [1]. The clustering of entities using ant system like algorithms leads to interesting solutions that obviously are the basis for further interactions as depicted in section 3. An example is shown in Fig 4. This figure represents an initial configuration with a random, uniform distribution of nodes.

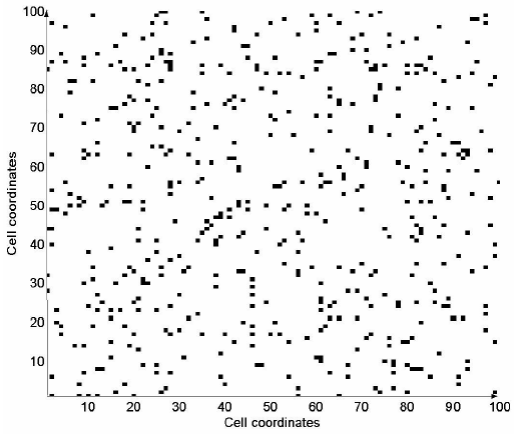


Fig 4. Clustering. Initial random distribution of nodes

Algorithms of swarm intelligence, i.e. collective ants, can be used to form clusters of nodes. In this example, only binary information is used to identify the relationship of nodes. Nevertheless, this algorithm can be enhanced to work in more sophisticated environments with multiple properties for an efficient clustering. A final state is shown in Fig 5.

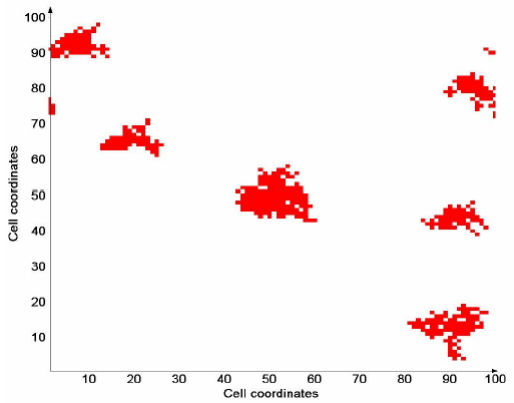


Fig 5. Cluster formation after running the algorithm some 10.000 steps [1]

After the successful group formation, other mechanisms can be applied for an efficient communication whether between individual systems (end-to-end, unicast), between all group members (group multicast), or between systems assigned a particular function (anycast).

5 Destinations and Goals

The objective of this paper is to provide information about mechanisms that allow us to build self-organizing network infrastructures. The key properties are efficiency and scalability. In order to address these issues, some bio-inspired mechanisms have been worked out: communication paradigms following the mechanisms in cell biology and clustering or group formation based on algorithms known from swarm intelligence. The primary destinations are discussed which point to novel solutions for various network environments employing the mentioned mechanisms.

Identification of properties

Instead of having a network administrator configuring individual systems and their properties, dynamic algorithms are required for self-organizing issues. In the past, such mechanisms were provided for identifying individual nodes based on an individual property, e.g. routing protocols are in some manner self-configuring by identifying neighboring nodes. Nevertheless, the final goal is to put new nodes into an existing network without any preconfigured knowledge. The properties of the nodes can be described in some common way. Bio-inspired communication mechanisms learnt from inter-cellular signaling pathways provides the appropriate mechanisms.

Localization of nodes

For some reasons, the position of special nodes must be detectable. In communication networks, typical examples are the localization of dedicated gateway nodes or the allocation of resources in mobile sensor networks. Swarm intelligence based clustering in combination with bio-inspired communications allows a fast and adaptive localization of resources even in unreliable network environments.

Group communication

One of the key features in autonomous systems is an efficient communication paradigm which connects multiple interacting nodes. This paradigm is called group communication. Research on such issues started with developments of IP multicast and is ongoing work in peer-to-peer networks. In most protocols, the management and maintenance of the group is the limiting factor in terms of scalability. Often, especially in pervasive environments or in sensor networks, group communication mechanisms are required which do not rely on the prior formation and maintenance of groups. Such communication paradigms can be derivated from cell biology. Here, fuzzy communication mechanisms are successfully employed showing a very high efficiency and specificity.

Task allocation
Modern task allocation mechanisms are related to distributed systems research. Nevertheless, many efforts were made using bio-inspired engineering approaches for better utilization of the global system. Interestingly, this is possible employing the discussed methodologies known from swarm intelligence in combination with new communication paradigms. The problem of task allocation can be reduced to the identification of available resources and group communication between related nodes.

Further work in the described destinations on application scenarios for bio-inspired networking methodologies is still required. Progress is currently driven by two factors. First, the analysis of biological mechanisms is looking for areas offering new paradigm and concepts which can be transferred to computer science related field. One example is the described application of cellular and molecular biology for communication networks. Secondly, some of the natural processes are still unknown or need further research in non-engineering field. We have to look constantly for emerging new achievements in science and adapt the changes to our algorithms.

6 Conclusions

We outlined biological inspired mechanisms for efficient and scalable communication in autonomous networking. Self-organization issues are promising to be the key answer to build large and complex systems fulfilling different tasks out of many simple independent autonomous entities. Such systems can be found quite often in computer networking. Network security, pervasive computing environments, and sensor networks are only single examples. All these systems require similar operations as their basis mechanisms:

- group formation
- adaptive communication
- resource localization and management

Mechanisms known from cell biology for the identification of resources and efficient inter-node communication in combination with swarm intelligence based approaches for adaptive group formation are adequate solutions for the depicted problems. In summary it can be said, that further developments of communication principles will have to rely on the elaborated methodologies in bio-inspired research.

Acknowledgements

This work is part of autonomous systems research at the chair for computer networks and communication systems, University of Erlangen-Nuremberg, headed by Prof. Dr. German. First studies on autonomous networking and self-organization issues were started in cooperation with Prof. Dr. Carle, chair for computer networks and internet at the University of Tuebingen. The research on bio-inspired networking is collaborative work with Dr. Krüger, Inst. of Physiology at the University of Erlangen-Nuremberg.

References

[1] Alenius, E., A. J. Eide, et al. (2004) Experiments on Clustering using Swarm Intelligence and Collective Behavior. International IPSI-2004 Stockholm Conference: Symposium on Challenges in the Internet and Interdisciplinary Research (IPSI-2004 Stockholm), Stockholm, Sweden.

[2] Bentley, P. J., T. Gordon, et al. (2001) New Trends in Evolutionary Computation. Congress on Evolutionary Computation (CEC-2001), Seoul, Korea.

[3] Bonabeau, E., M. Dorigo, et al. (1999) Swarm Intelligence: From Natural to Artificial Systems, Oxford University Press.

[4] Camazine, S., J.-L. Deneubourg, et al. (2003) Self-Organization in Biological Systems, Princeton University Press.

[5] Chahl, J., S. Thakoor, et al. (2003) Bioinspired Engineering of Exploration Systems: A Horizon/Attitude Reference System Based on the Dragonfly Ocelli for Mars Exploration Applications. Journal of Robotic Systems 20(1): 35-42.

[6] Das, S. K., N. Banerjee, et al. (2004) Solving Otimization Problems in Wireless Networks using Genetic Algorithms. Handbook of Bio-inspired Algorithms.

[7] D'haeseleer, P., S. Forrest, et al. (1996) An Immunological Approach to Change Detection: Algorithms, Analysis and Implications. IEEE Symposium on Security and Privacy, Oakland, CA, USA.

[8] Di Caro, G. and M. Dorgio (1998) AntNet: Distributed Stigmergetic Control for Communication Networks. Journal of Artificial Intelligence Research 9: 317-365.

[9] Dorigo, M., V. Maniezzo, et al. (1996) The Ant System: Optimization by a colony of cooperating agents. IEEE Transactions on Systems, Man, and Cybernetics 26(1): 1-13.

[10] Dressler, F. (2004) Bio-inspired mechanisms for efficient and adaptive network security mechanisms. Dagstuhl Seminar 04411 on Service Management and Self-Organization in IP-based Networks, Schloss Dagstuhl, Wadern, Germany.

[11] Dressler, F. (2004) Bio-inspirierte effiziente Datenkommunikation in mobilen Netzen. Systemsoftware für Pervasive Computing: Fachgespräch der GI/ITG-Fachgruppe Kommunikation und Verteilte Systeme, Stuttgart, Germany.

[12] Dressler, F. and B. Krüger (2004) Cell biology as a key to computer networking. Bielefeld, Germany, German Conference on Bioinformatics 2004 (GCB'04).

[13] Eigen, M. (1979) The Hypercycle: A Principle of Natural Self Organization, Springer Verlag.

[14] Gerhenson, C. and F. Heylighen (2003) When Can we Call a System Self-organizing? 7th European Conference on Advances in Artificial Life (ECAL 2003), Dortmund, Germany.

[15] Hofmeyer, S. (1999) An Immunological Model of Distributed Detection and Its Application to Computer Security, University of New Mexico.

[16] Hofmeyer, S. and S. Forrest (2000) Architecture for an Artificial Immune System. Evolutionary Computation 8(4): 443-473.

[17] Kadrovach, B. A. and G. B. Lamont (2002) A Practicle Swarm Model for Swarm-based Networked Sensor Systems. 2002 ACM Symposium on Applied Computing, Madrid, Spain.

[18] Kennedy, J. and R. Eberhart (2001) Swarm Intelligence, Morgan Kaufmann.

[19] Kephart, J. O. (1994) A Biologically Inspired Immune System for Computers. 4th International Workshop on Synthesis and Simulation of Living Systems, Cambridge, Massachusetts, USA, MIT Press.

[20] Krüger, B. and F. Dressler (2004) Molecular Processes as a Basis for Autonomous Networking. International IPSI-2004 Stockholm Conference: Symposium on Challenges in the Internet and Interdisciplinary Research (IPSI-2004 Stockholm), Stockholm, Sweden.

[21] Le Boudec, J.-Y. and S. Sarafijanovic (2004) An Artificial Immune System Approach to Misbehavior Detection in Mobile Ad-Hoc Networks. First International Workshop on Biologically Inspired Approaches to Advanced Information Technology (Bio-ADIT2004), Lausanne, Switzerland.

[22] Liang, Y.-C. and A. E. Smith (1999) An ant system approach to redundancy allocation. 1999 Congress on Evolutionary Computation, Washington D.C.

[23] Muraleedharan, R. and L. A. Osadiciw (2003) Balancing The Performance of a Sensor Network Using an Ant System. 37th Annual Conference on Information Sciences and Systems (CISS 2003), Baltimore, MD.

[24] Muraleedharan, R. and L. A. Osadiciw (2003) Sensor Communication Network Using Swarm Intelligence. 2nd IEEE Upstate New York Workshop on Sensor Networks, Syracuse, NY, USA.

[25] Muraleedharan, R. and L. A. Osadiciw (2004) A Predictive Sensor Network Using Ant System. Defense and Security Symposium, Orlando, Florida, USA.

[26] Song, S. and T. Suda (2001) Security on Energy Level in the Bio-Networking Architecture. 3rd International Conference on Advanced Communication Technology (ICACT2001), Muju Resort, South Korea.

[27] Suzuki, J. and T. Suda (2003) The Bio-Networking Platform: An Autonomic Agent Platform for Pervasive Computing. 2nd IPSJ Workshop on Ubiquitous Computing, Kyoto, Japan.

[28] Suzuki, J. and T. Suda (2003) Design and Implementation of a Scalable Infrastructure for Autonomous Adaptive Agents. 15th IASTED International Conference on Parallel and Distributed Computing and Systems, Marina del Ray, CA, USA.

[29] Thakoor, S., J. Chahl, et al. (2004) BEES: Exploring Mars with Bioinspired Technologies. IEEE Computer 37(9): 38-47.

[30] Wang, M. and T. Suda (2001) The Bio-Networking Architecture: A Biologically Inspired Approach to the Design of Scalable, Adaptive, and Survivable/Available Network Applications. 1st IEEE Symposium on Applications and the Internet (SAINT), San Diego, CA, USA.

Model Checking Multi-Agent Systems

Mustapha Bourahla
Computer Science Department, University of Biskra, Algeria
mbourahla@hotmail.com

Mohamed Benmohamed
Computer Science Department, University of Constantine, Algeria
ibnm@yahoo.fr

Keywords: Multi-agent systems, Multi-modal branching-timelogic, Model checking

Received: September 21, 2004

Multi-agent systems are increasingly complex, and the problem of their verification and validation is acquiring increasing importance. In this paper we show how a well known and effective verification technique, model checking, can be generalized to deal with multi-agent systems. This paper explores a particular type of multi-agent system, in which each agent is viewed as having the three mental attitudes of belief (B), desire (D), and intention (I). We present a new approach to the verification of multi-agent systems, based on the use of possible-worlds framework to describe the system, a multi-modal branching-time logic BDI_{CTL}, with a semantics that is grounded in traditional decision theory, to specify the properties, and a decision procedure based on model checking technique. An imperative multi-agent programming language and a formal semantics for this language in terms of the BDI_{CTL} logic are used to specify multi-agent systems. The multi-agent program is used to systemically construct the agents state spaces. Then an automatic synthesis of these state spaces using the agents mental attitudes will generate the possible worlds structures. These possible worlds will be used by the adopted decision procedure to solve the problems of verification. A preliminary implementation of the approach shows promising results.

Povzetek: Predstavljen je nov algoritem za preverjanje pravilnosti multi-agentnih sistemov.

1 Introduction

The design of (in particular safety-critical control) systems that are required to perform high-level management and control tasks in complex dynamic environments is becoming of increasing commercial importance. Such systems include the management and control of air traffic systems, telecommunications networks, business processes, space vehicles, and medical services. Experience in applying conventional software techniques to develop such systems has shown that they are very difficult and very expensive to build, verify, and maintain. Agent-oriented systems, based on a radically different view of computational entities, offer prospects for a qualitative change in this position.

A number of different approaches have emerged as candidates for the study of agent-oriented systems [3, 6, 16, 18, 19]. One such architecture [16] views the system as a rational agent having certain mental attitudes of Belief (beliefs can be viewed as the informative component of system state), Desire (desires can be thought of as representing the motivational state of the system), and Intention (the intentions of the system capture the deliberative component of the system). Thus BDI (Belief, Desire and Intention) represents the information, motivational, and deliberative states of the agent. These mental attitudes determine the system's behavior and are critical for achieving adequate or optimal performance when deliberation is subject to resource.

To describe the belief, desire, and intention components of the system state a propositional form is used, based on possible worlds. Thus, the possible worlds model [16] consists of a set of possible worlds where each possible world is a tree structure. A particular index within a possible world is called a situation. With each situation we associate a set of belief-accessible worlds, desire-accessible worlds, and intention-accessible worlds; intuitively, those worlds that the agent believes to be possible, desires and intends to bring about, respectively.

In this paper, we address the problem of verification for such formalisms which is increasingly important. The formalism of multi-agent temporal logic [16] is introduced towards lifting one of the most successful verification techniques, model checking [4], for the validation of multi-agent systems. Multi-agent temporal logic BDI_{CTL} combines, within a single framework, the aspects of temporal logic, used to reason about the temporal evolution of finite-state automata, with agent-related aspects such as belief, desire and intention.

The problem of extending the standard temporal logic model checking techniques, and then using the related

tools, to deal with the multi-agent aspects of the logic, is the specification of the possible worlds and the relation between them. The essential of our contribution is to present an approach by which we help reducing the specification time. This approach is based on the automatic synthesis of the mental attitudes of agents. Each mental state will be an index to a new created world using the specifications of the different agents. For illustrating our approach, we designed a sub-language for specifying multi-agent systems. The specification will be agent-oriented. A tool is developed for constructing the state space of each agent in the multi-agent system. Then an algorithm is developed for synthesizing the agent models of the specified multi-agent system. The synthesis result is a possible worlds model. At the end, we have adopted the standard model checking for the analysis of these models of multi-agent systems. A symbolic model checking tool for verifying multi-agent systems has been implemented. The preliminary results are extremely promising.

This paper is structured as follows. In Section 2 we describe the multi-agent temporal logic (BDI_{CTL}). In Section 3, we present the specification sub-language and its underlying intuitions, and define the language and the semantics as a temporal logic. In Section 4, we present the algorithm for synthesizing the corresponding multi-agent structures. In Section 5, we present the extended general algorithm for model checking. Finally, in Section 6 we outline the results, discuss future work, and draw some conclusions.

2 Multi-Agent Temporal Logic BDI_{CTL}

The temporal logic BDI_{CTL} [16] we consider is extension of Computation Tree Logic CTL [7] that has been used extensively for reasoning about concurrent programs. The branching-time logic CTL is extended to represent the mental state or belief-desire-intention state of an agent. This logic can then be used to reason about agents and the way in which their beliefs, desires, and actions can bring about the satisfaction of their desires. The syntax of BDI_{CTL} is as follows.

$$\varphi ::= true \mid p \mid \neg\varphi \mid \varphi \vee \varphi \mid \exists X\varphi \mid \exists G\varphi \mid \exists \varphi U \varphi \mid B_i\varphi \mid D_i\varphi \mid I_i\varphi.$$

The primitives of this language include a nonempty set AP of atomic propositions, propositional connectives $\vee$ and $\neg$, modal operators B (agent believes), D (agent desires), and I (agent intends), and temporal operators of CTL. The CTL temporal operators are $\exists X\varphi$ (φ might hold at next time instant), $\exists \varphi U \psi$ (it might be the case that ψ holds at a certain time future and until then φ holds), and $\exists G\varphi$ (φ might hold for all future time instants). Temporal operators are compactly characterized by $\exists \varphi U \psi \Leftrightarrow (\psi \vee (\varphi \wedge \exists X \exists (\varphi U \psi)))$ and by $\exists G\varphi \Leftrightarrow (\varphi \wedge \exists X \exists G\varphi)$. We have operators $B_i\varphi$, $D_i\varphi$, and $I_i\varphi$ which mean that agent i has a belief, desire, and intention of φ, respectively. This grammar is not given in its most succinct form and there exist equivalence rules to express the same formula with different operators; for example, $\forall F\varphi$ (φ is inevitable) is equivalent to $\neg\exists G\neg\varphi$. In practice, by using these equivalence rules, a formula can be written such that the negation appears only at the level of atomic propositions. Such a form of a formula is known as Negative Normal Form (henceforth NNF form).

The traditional possible-worlds semantics of beliefs considers each world to be a collection of propositions and models belief by a belief-accessibility relation $\mathcal{B}$ linking these worlds. A formula is said to be believed in a world if and only if it is true in all its belief-accessible worlds [10]. The accessibility relation $\mathcal{B}$ is a relation between the world at an index and at a time point to a set of worlds. Intuitively, an agent believes a formula in a world at a particular index if and only if in all its belief-accessible worlds the formula is true. We consider each possible world to be a tree structure with a single past and a branching future [5]. Evaluation of formulas is with respect to a world and a state. Hence, a state acts as an index into a particular tree structure or world of the agent. The belief-accessibility relation maps a possible world at a state to other possible worlds. The desire-, and intention-accessibility relations behave in a similar fashion. More formally, we have the following definition of a Kripke structure.

Definition 1 *A Kripke structure is defined to be a tuple* $K = \langle W, S, \{S_w : w \in W\}, \{R_w : w \in W\}, \{I_w : w \in W\}, L, \mathcal{B}, \mathcal{D}, \mathcal{I}\rangle$, *where* W *is a set of possible worlds,* S *is the set of states,* S_w *is the set of states in each world* $w \in W$ *(*$S = \cup_{w\in W} S_w$*),* R_w *is a total tree relation, i.e.,* $R_w \subseteq S_w \times S_w$, I_w *a set of initial states (*$I_w \subseteq S_w$*),* $L : W \times S \rightarrow 2^{AP}$ *is a function that labels for each world* $w \in W$*, each state* $s \in S_w$ *with the set of atomic propositions true in that state, and* $\mathcal{B}$*,* $\mathcal{D}$*, and* $\mathcal{I}$ *are relations on the worlds* W *and states* S *(i.e.* $\mathcal{O} \subseteq W \times S \times W$*), where* $\mathcal{O}$ *is one of* $\mathcal{B}$*,* $\mathcal{D}$*, or* $\mathcal{I}$*.*

We also define a world to be a sub-world of another if one of them contains fewer paths, but they are otherwise identical to each other. More formally, we have the following definition.

Definition 2 *A world* w' *is a sub-world of the world* w*, denoted by* $w' \sqsubseteq w$*, if and only if*

1. $S_{w'} \subseteq S_w$, $I_{w'} \subseteq I_w$, $R_{w'} \subseteq R_w$,
2. $\forall s \in S_{w'}$, $L(w', s) = L(w, s)$,
3. $\forall s \in S_{w'}$, $(w', s, v) \in \mathcal{B}$ *iff* $(w, s, v) \in \mathcal{B}$*; and similarly for* $\mathcal{D}$ *and* $\mathcal{I}$*.*

The semantics of BDI_{CTL} involves two dimensions: an epistemic and a temporal dimension. The truth of a formula depends on both the epistemic world w and the temporal state s. A pair (w, s) (denoted also s^w) is called a situation

in which BDI_{CTL} formulas are evaluated. The relation between situations is traditionally called an accessibility relation (for beliefs) or a successor relation (for time).

A BDI_{CTL}-model $\mathcal{M}$ is represented as a Kripke structure. We note a model $\mathcal{M}$ in world w as $\mathcal{M}_w$. A trace (path) in a world $w \in W$ starting from s^w is an infinite sequence of states $\rho_w = s_0^w s_1^w s_2^w \cdots$ such that $s_0^w = s^w$, and for every $i \geq 0$, $\langle s_i^w, s_{i+1}^w \rangle \in R_w$. The $(i+1)$-th state of trace ρ_w is denoted $\rho_w[i]$. The set of paths starting in state s^w of the model $\mathcal{M}_w$ is defined by $\Pi_{\mathcal{M}_w}(s^w) = \{\rho_w \mid \rho_w[0] = s^w\}$.

For any BDI_{CTL}-model $\mathcal{M}_w$ and state $s^w \in S_w$, there is an infinite computation tree with root labeled s^w such that $\langle s_i^w, s_j^w \rangle$ is an arc in the tree if and only if $\langle s_i^w, s_j^w \rangle \in R_w$. Satisfaction of formulas, denoted by $\models_{\mathcal{M}_w}$, is given with respect to a model $\mathcal{M}$, a world w, and state s. The expression $s \models_{\mathcal{M}_w} \varphi$ is read as "model $\mathcal{M}$ in world w and state s satisfies φ".

- $s \models_{\mathcal{M}_w} p$ iff $p \in L(w, s)$
- $s \models_{\mathcal{M}_w} \neg p$ iff $s \not\models_{\mathcal{M}_w} p$
- $s \models_{\mathcal{M}_w} \varphi \vee \psi$ iff $s \models_{\mathcal{M}_w} \varphi$ or $s \models_{\mathcal{M}_w} \psi$
- $s \models_{\mathcal{M}_w} \exists X\varphi$ iff $\exists \rho_w \in \Pi_{\mathcal{M}_w}(s) : \rho_w[1] \models_{\mathcal{M}_w} \varphi$
- $s \models_{\mathcal{M}_w} \exists G\varphi$ iff $\exists \rho_w \in \Pi_{\mathcal{M}_w}(s) : \forall j \geq 0 : \rho_w[j] \models_{\mathcal{M}_w} \varphi$
- $s \models_{\mathcal{M}_w} \exists \varphi U \psi$ iff $\exists \rho_w \in \Pi_{\mathcal{M}_w}(s) : (\exists j \geq 0 : \rho_w[j] \models_{\mathcal{M}_w} \psi) \wedge (\forall k, 0 \leq k < j : \rho_w[k] \models_{\mathcal{M}_w} \varphi)$
- $s \models_{\mathcal{M}_w} B_i(\varphi)$ iff $\forall v, (w, s, v) \in \mathcal{B} : \forall s' \in \mathcal{B}(B_i(\varphi)) : s' \models_{\mathcal{M}_v} \varphi$
- $s \models_{\mathcal{M}_w} D_i(\varphi)$ iff $\forall v, (w, s, v) \in \mathcal{D} : \forall s' \in \mathcal{D}(D_i(\varphi)) : s' \models_{\mathcal{M}_v} \varphi$
- $s \models_{\mathcal{M}_w} I_i(\varphi)$ iff $\forall v, (w, s, v) \in \mathcal{I} : \forall s' \in \mathcal{I}(I_i(\varphi)) : s' \models_{\mathcal{M}_v} \varphi$

We denote the set of states in the world v that are accessible to the state s in the world w, where $(w, s, v) \in \mathcal{O}$ by $\mathcal{O}(O_i(\varphi))$ (more details are in Section 4). A formula φ is said to be valid in $\mathcal{M}_v$, written as $\models_{\mathcal{M}_v} \varphi$, if $s \models_{\mathcal{M}_v} \varphi$ for every state $s \in S_v$. A formula is valid if it is true in every state, in every world, in every structure (model).

Recall that an agent i has a belief φ, denoted $B_i(\varphi)$, in state s if and only if φ is true in all the belief-accessible worlds of the agent at state s. As the belief-accessibility relation is dependent on the state, the mapping of B at some other state may be different. Thus the agent can change its beliefs about the options available to it. Similar to belief-accessible worlds, for each state we also associate a set of desire-accessible worlds to represent the desires of the agent. Thus, in the same way that we treat belief, we say that the agent has a desire φ in state s if and only if φ is true in all the desire-accessible worlds of the agent in state s.

In the philosophical literature, desires can be inconsistent and the agent need not know the means of achieving these desires. Desires have the tendency to 'tug' the agent in different directions. They are inputs to the agent's deliberation process, which results in the agent choosing a subset of desires that are both consistent and achievable. In the AI literature such consistent achievable desires are usually called goals. The desires as presented here are logically consistent, but due to the branching-time structure, conflicting desires can 'tug' the agent along different execution paths. That is, while the desires may be logically consistent, they may not all be realizable, as the agent can only follow one execution path in the branching tree of possible executions. The deliberation process must eventually resolve these conflicts and choose a set of realizable desires before the agent can act intentionally.

Intentions are similarly represented by sets of intention-accessible worlds. These worlds are ones that the agent has chosen to attempt to realize. The intention-accessibility relation is used to map the agent's current world and state to all its intention-accessible worlds. We say that the agent intends a formula in a certain state if and only if it is true in all the agent's intention-accessible worlds at that state.

3 Specification of Multi-Agent Systems

A multi-agent system contains a finite number of agents. The basic form of an agent is "agent A is init P", where A is the name of the agent and P is the program body. Each agent in a multi-agent system is assumed to have a unique name, drawn from a set of $agent\ identifiers$. The main part of an agent, which determines its behavior, is the program body P. The basis of program bodies is a simple imperative language, containing iteration ($loop$ loops), sequence (the ; constructor), selection (a form of the if, $then$, $else$ statement), choice (the | constructor), and assignment statements.

An agent A is allowed to execute by a do instruction any of a set $Actions = \{\alpha, \cdots\}$ of $external\ actions$. The simplest way to think of external actions is as $native\ methods$ in a programming language like Java. They provide a way for agents to execute actions that do not simply affect the agent's internal state, but its external environment. The basic form of the do instruction is $do\ \alpha$, where $\alpha \in Actions$ is the external action to be performed. When we incorporate communication, we do so by modeling message sending as an external action to be performed.

In a conventional programming language, conditions in if statement are only allowed to be dependent on program variables. Unusually, we allow conditions in if statement to be arbitrary formulas of the BDI_{CTL} logic (any acceptable formula is allowed as a condition). To make this more concrete, consider the following:

$$if\ B_j p\ then\ r\ :=\ p\ else\ r := false$$

The idea is that if the agent executing this instruction believes that agent j believes that p, then the agent executing the instruction assigns the value of p to r. If the agent executing the instruction believes it is not the case that agent j believes p, then it assigns the value $false$ to r. Notice

the form of words used here: the agent executing this if instruction must believe that j believes p; the condition does not depend on what j actually believes, but on what the agent executing the statement believes that j believes. As this example illustrates, conditions can thus refer to the mental state of other agents. The general form of a $loop$ construct, as in conventional programming languages, is $loop\ P\ endloop$, where P is a program.

Given a collection $\{A_1, \cdots, A_n\}$ of agents, they are composed into a multi-agent system by the parallel composition operator "$\|$": $A_1 \parallel \cdots \parallel A_n$. Formally, the abstract syntax of multi-agent systems is defined by the grammar below.

MAS ::= $Agent \parallel \cdots \parallel Agent$
Agent ::= agent A is Init P
Init ::= init $p := true\ or\ false$, where $p \in AP$
P ::= *do* $\alpha \mid p := true\ or\ false$
$\mid if\ \varphi\ then\ P \mid if\ \varphi\ then\ P\ else\ P$
$\mid loop\ P\ endloop \mid P\ ';'\ P \mid P\ '|'\ P$

Example 1 *To clarify this syntax, let us consider the following scenario involving two agents: a receiver rcv and a sender snd. snd continuously reads news on a certain subject (p) from its sensors (e.g., the standard input). Once read the news, snd informs rcv only if it believes that rcv does not have the correct knowledge about that subject (this in order to minimize the traffic over the network). Once received the news, rcv acknowledges this fact back to snd. After the reception of acknowledgement from the agent rcv, the agent snd will believe that the agent rcv believes $p \vee q$ (q is a propositional atom) or it believes that the agent rcv believes p (or q) in the case that the agent rcv at the beginning, does not have the correct knowledge about p (or $\neg p$).*

We have therefore three agents: snd, rcv, and a network (communication protocol) $protocol$ which allows them to interact. The descriptions of snd, rcv and the communication protocol $protocol$, are given below respectively.

agent snd is
init $\forall p \in AP : p := false$
loop
do read(p);
if $p \wedge \neg B_{rcv} p$ *then*
do putmsg(inform(snd, rcv, p));
do getmsg(inform(rcv, snd, $B_{rcv} p$));
$(B_{rcv} p := true) \mid (B_{rcv}(p \vee q) := true)$;
else if $\neg p \wedge \neg B_{rcv} \neg p$ *then*
do putmsg(inform(snd, rcv, $\neg p$));
do getmsg(inform(rcv, snd, $B_{rcv} \neg p$));
$(B_{rcv} q := true) \mid (B_{rcv}(p \vee q) := true)$;
endloop

agent rcv is
init $\forall p \in AP : p := false$
loop
{
do getmsg(inform(snd, rcv, p));
$p := true$;
do putmsg(inform(rcv, snd, $B_{rcv} p$));
} | {
do getmsg(inform(snd, rcv, $\neg p$));
$q := true$;
do putmsg(inform(rcv, snd, $B_{rcv} \neg p$));
}
endloop

agent $protocol$ is
init $\forall p \in AP : p := false$
loop
$\forall p \in AP : p := false$;
{
$B_{snd} \forall F$ *do putmsg(inform(snd, rcv, p)) :=* $true$;
$B_{rcv} \forall F$ *do getmsg(inform(snd, rcv, p)) :=* $true$
} | {
$B_{snd} \forall F$ *do putmsg(inform(snd, rcv,* $\neg p$*)) :=* $true$;
$B_{rcv} \forall F$ *do getmsg(inform(snd, rcv,* $\neg p$*)) :=* $true$
};
$\forall p \in AP : p := false$;
{
$B_{rcv} \forall F$ *do putmsg(inform(rcv, snd,* $B_{rcv} p$*)) :=* $true$;
$B_{snd} \forall F$ *do getmsg(inform(rcv, snd,* $B_{rcv} p$*)) :=* $true$
} | {
$B_{rcv} \forall F$ *do putmsg(inform(rcv, snd,* $B_{rcv} \neg p$*)) :=* $true$;
$B_{snd} \forall F$ *do getmsg(inform(rcv, snd,* $B_{rcv} \neg p$*)) :=* $true$
}
endloop

$mas = protocol \parallel snd \parallel rcv$

In these descriptions, the news subject of the information exchange is the truth value of the propositional atom p. $inform(snd, rcv, p)$ returns a message with sender snd, receiver rcv, and content p ($inform$ is a FIPA (Foundation for Intelligent Physical Agents) primitive). $putmsg$ and $getmsg$ are the primitives for putting and getting (from the communication channel) a message. $read$ allows for reading from the standard input. B_{rcv} is the operator used to represent the beliefs of rcv as perceived by the other agents, and dually for B_{snd}. Notice that the communication protocol has beliefs about rcv and snd and therefore must have a representation of how they behave. We suppose that this representation coincides with what rcv and snd actually are, as described above. This allows us to model the fact that the communication protocol behaves correctly following what snd and rcv do. snd also has beliefs about rcv. We suppose that snd (which in principle does not know anything about how rcv works) only knows that rcv can be in one of two states, with p being either $true$ or $false$. In the example, $B_{snd} \forall F\ do < statement >$ (or $B_{rcv} \forall F\ do < statement >$) intuitively means that $snd(rcv)$ will necessarily reach a state in which it will have just performed the action corresponding to $< statement >$. The agent program $protocol$ codifies the fact that the protocol implements the information flow between snd and rcv, and the fact that it always delivers the messages it is asked to deliver. Some properties that we may want to prove are:

1. *An agent liveness property, e.g., that snd will eventually believe that rcv believes p or believes $\neg p$. Its expression is $\models_{\mathcal{M}_{w_{snd}}} \forall F(B_{rcv} p \vee B_{rcv} \neg p)$. Where w_{snd} is the world seen by the agent snd.*

2. *An overall system liveness property, e.g., that if it believes p, then in the future snd will believe that rcv*

will believe p. Its expression is $\models_{\mathcal{M}} B_{snd}(p) \supset \forall F B_{snd} \forall F B_{rcv} p$.

3.1 Formal Semantics

The semantics of a multi-agent program will be defined as a formula of BDI_{CTL}, which characterizes the acceptable computations of the system, and the "mental state" of the agents in the system.

The agent program semantic function is defined in terms of the function $[\![\cdots]\!]_{Bexp} : Bexp \rightarrow B$, which gives the semantics of Boolean expressions. The four remaining semantic functions are defined in Figure 1. The idea is that the semantics are defined inductively by a set of definitions, one for each construct in the language.

A declaration "$agent\ A\ is\ init\ P$" binds a name A with the semantics of the $init$ statements and the program body P. We capture the semantics of this by systematically substituting name A for the place-holder name $self$ in $[\![init]\!]_{Init} \wedge [\![P]\!]_P$. The semantics of a system $A_1 \parallel \cdots \parallel A_n$ is simply the conjunction of the semantics of the component agents A_i, together with some background assumptions ψ_{MAS}. The idea of the background assumptions is that these capture general properties of a multi-agent system that are not captured by the semantics of the language.

4 Construction of Possible Worlds Model

We will develop an algorithm to construct a multi-agent structure as defined in Definition 1. First we need to build a structure for each agent specification then we will synthesize these structures. At the beginning, a multi-agent system will have a Kripke structure of the form $K = \langle W = \{w_1, \cdots, w_n\}, S = \{S_{w_1}, \cdots, S_{w_n}\}, R = \{R_{w_1}, \cdots, R_{w_n}\}, I = \{I_{w_1}, \cdots, I_{w_n}\}, L, \mathcal{B} = \emptyset, \mathcal{D} = \emptyset, \mathcal{I} = \emptyset\rangle$, where n is the number of agents. Then we will compute the sets $\mathcal{B}$, $\mathcal{D}$, and $\mathcal{I}$ using the worlds $w \in W$ and the labeling function L. At the end, a Kripke structure K will be constructed representing the multi-agent system using the algorithm below. The initial Kripke structure K is generated directly from the agents specifications. In each world, there is a finite set of the BDI operators of the form $O_i\varphi$ (where O stands for B, D, or I). This set is considered as a part of the atomic propositions AP.

Let us call $TrueBDI_{(w,v)}(s)$ the set of BDI atoms of world w (of the current agent), of the form $O_i\varphi$, which are true at s ($TrueBDI_{(w,v)}(s) = BDI_{(w,v)} \cap L(w,s)$). v is the world of the agent i. An accessibility relation $\mathcal{O}_{(w,v)} \subseteq BDI_{(w,v)} \times S_v$ (or $\mathcal{O}_{(w,v)}(O_i\varphi) \subseteq S_v$), constraints the truth of BDI ($O_i\varphi$) atoms of a world w to the truth values (of φ) in the world v. The states of world v accessible to s are those states belonging to the intersection, over the BDI atoms true at s, of the sets of states accessible to $TrueBDI_{(w,v)}(s)$. We extend the accessiblity relation to a relation over a set of BDI atoms $\mathcal{A} \subseteq BDI_{(w,v)}$ as follows.

$$\mathcal{O}_{(w,v)}(\mathcal{A}) = \bigcap_{O_i\varphi \in \mathcal{A}} \mathcal{O}_{(w,v)}(O_i\varphi)$$

Therefore, the set of states of v accessible to a state s of w will be simply denoted by $\mathcal{O}_{(w,v)}(TrueBDI_{(w,v)}(s))$.

Depending on the kind of BDI operator being considered, the accessibility relation may have different properties. What makes $\mathcal{M}$ a model of a multi-agent possible world is the particular structure of the accessibility relations among adjacent sub-worlds.

Definition 3 *A BDI_{CTL} model $\mathcal{M}$ is a possible world structure if for every word w, every BDI atom $O_i\varphi$ of w and every $s \in S_w$ the following conditions hold.*

1. *If $O_i\varphi \in L(w,s)$, then $s' \in \mathcal{O}_{(w,v)}(TrueBDI_{(w,v)}(s))$ implies that s' is reachable in v and $s' \models_{\mathcal{M}_v} \varphi$.*
2. *If $O_i\varphi \notin L(w,s)$, then for some reachable state $s' \in \mathcal{O}_{(w,v)}(TrueBDI_{(w,v)}(s))$, $s' \models_{\mathcal{M}_v} \neg\varphi$.*

Condition 1 tells us what are the states in world v which are accessible to a given state s (satisfying $TrueBDI_{(w,v)}(s)$), according to the semantics of $BDIs$, namely that the argument of a BDI true at a state must be true in all the states reachable from it via accessibility relation. Condition 2, on the other hand, tells us what are the states of world w which actually comply to the semantics of $BDIs$, i.e. the states which assign truth values to BDI atoms in accordance with the semantics of the BDI operator.

Let φ and ψ be two BDI_{CTL} formulas, assume that $\varphi \supset \psi$, it would be unreasonable to allow for a state satisfying the BDI atom $O_i\varphi$, yet not satisfying the BDI atom $O_i\psi$ at the same time. This is the kind of situation that this condition prevents. Indeed, let us suppose there is a state s of a world w satisfying $O_i\varphi$. By Condition 1 of Definition 3, any reachable state s' of world v accessible to s must satisfy φ. By Condition 2 of Definition 3, for s not to satisfy the BDI atom $O_i\psi$, there must be a reachable state s'' in world v accessible to s and which does not satisfy ψ. But, according to Condition 1, all the states accessible to s must satisfy φ and, consequently, ψ as well, which is impossible.

On the other hand, the definition of multi-agent structure allows for a state s of a world w to satisfy both BDI atoms $O_i\varphi$ and $O_i\neg\varphi$, where φ is BDI_{CTL} formula. This happens when there is no state in world v is accessible to s (i.e. when $\mathcal{O}_{(w,v)}(TrueBDI_{(w,v)}(s))$ is empty). This corresponds to the situation where world w, when in state s, ascribes inconsistent $BDIs$ to world v. Notice however that this kind of inconsistency is of a different nature from the one ruled out by Definition 3. Indeed, allowing a state s not to satisfy $O_i\psi$ while satisfying $O_i\varphi$ (where $\varphi \supset \psi$) would make the specification of w itself inconsistent, while allowing both $O_i\varphi$ and $O_i\neg\varphi$ would not. It

$$
\begin{aligned}
[\![init\ p]\!]_{Init} &= B_{self}p, p \in AP \\
[\![do\ \alpha]\!]_P &= I_{self}\alpha, \alpha \in Actions \\
[\![p := e]\!]_P &= \forall X B_{self}[\![e]\!]_{Bexp} \\
[\![if\ \varphi\ then\ P]\!]_P &= B_{self}\varphi \Rightarrow [\![P]\!]_P \\
[\![if\ \varphi\ then\ P_1\ else\ P_2]\!]_P &= B_{self}\varphi \Rightarrow [\![P_1]\!]_P \wedge \neg B_{self}\varphi \Rightarrow [\![P_2]\!]_P \\
[\![loop\ P\ endloop]\!]_P &= [\![P\ ;\ loop\ P\ endloop]\!]_P \\
[\![P_1; P_2]\!]_P &= [\![P_1]\!]_P \Rightarrow [\![P_2]\!]_P \\
[\![P_1 \mid P_2]\!]_P &= [\![P_1]\!]_P \vee [\![P_2]\!]_P \\
[\![agent\ A\ is\ init\ P]\!]_{Agent} &= ([\![init]\!]_{Init} \wedge [\![P]\!]_P)[A \mapsto self] \\
[\![A_1 \parallel \cdots \parallel A_n]\!]_{MAS} &= [\![A_1]\!]_{Agent} \wedge \cdots \wedge [\![A_n]\!]_{Agent} \wedge \psi_{MAS}
\end{aligned}
$$

Figure 1: Semantics of multi-agent program

is clearly possible, though, to rule out also the latter situation, by adding the additional constraint that every state s must have a non-empty set of accessible states of the world below (i.e. $\mathcal{O}_{(w,v)}(TrueBDI_{(w,v)}(s)) \neq \emptyset$).

4.1 Synthesizing Multi-Agent Structure

In this section we present a synthesis algorithm that automatically constructs the suitable multi-agent Kripke structure $\mathcal{M}$ from a set of independently generated structures for each agent specification and a selected set of BDI atoms, thus leading to significant savings in the modeling phase. The synthesis algorithm is reported below. It takes as input a set of agents represented as world structures, and a set of BDI atoms. Intuitively, the algorithm at each world computes as a first step the accessibility relations associated to each BDI operator of the world. This is done according to Condition 1 of Definition 3. The second step is to implement Condition 2 of the same definition. The idea is to check whether there are states of the current world where the negation of some BDI atoms conflicts with other BDI atoms true at that state. Condition 2 tells us no such state is admissible in a multi-agent structure as they correspond to impossible combination of BDI atoms. Therefore, we need to get rid of all those states in the structure of the world. Once those two steps are performed at each world, the resulting structure is indeed a multi-agent structure.

Algorithm 1 *BUILD-MODEL(w, $\mathcal{M}$)*

{
for each $i \in agent\ identifiers$ do
Let v be the world structure of the agent i
if $BDI_{(w,v)} \neq \emptyset$ then
Let wv be the world of the agent i as viewed by the agent of the world w
$\mathcal{M} \leftarrow$ BUILD-MODEL(wv, $\mathcal{M}$)
$\mathcal{M} \leftarrow$ CreateAR(w, v, $\mathcal{M}$)
end if
end for
return($\mathcal{M}$)
}

The initial call is BUILD-MODEL(top, $\mathcal{M}$), where top is the root of the Kripke structure (in our example, is the $protocol$ agent). At the end of the algorithm, $\mathcal{M}$ will contain the accessibility relations of the structure rooted at w. The algorithm BUILD-MODEL recursively descends depth-first the tree of worlds rooted at w, and builds the accessibility relations (algorithm below) with all the worlds one level below the current world w. The creation of the accessibility relations is using the algorithm MAS-Sat(w, φ) (descried in the next section) which computes the set of states satisfying the formula φ in the world w.

Algorithm 2 *CreateAR(w, v, $\mathcal{M}$)*

{
/ Condition 1 of Definition 3 */*
for each $O_i\varphi \in BDI_{(w,v)}$ do
$[\![\varphi]\!]_v \leftarrow$ MAS-Sat(v, φ)
$\mathcal{O}_{(w,v)}(O_i\varphi) \leftarrow [\![\varphi]\!]_v$
end for
/ Condition 2 of Definition 3 */*
$BadStates \leftarrow \emptyset$
for each $O_i\varphi \in BDI_{(w,v)}$ do
$[\![\neg\varphi]\!]_v \leftarrow$ MAS-Sat(v, $\neg\varphi$)
$BadBDI \leftarrow \{\mathcal{A} \subseteq BDI_{(w,v)} \setminus \{O_i\varphi\} \mid \mathcal{O}_{(w,v)}(\mathcal{A}) \cap [\![\neg\varphi]\!]_v = \emptyset\}$
$BadStates \leftarrow BadStates \cup \{s \in S_w \mid TrueBDI_{(w,v)}(s) \subseteq BadBDI\}$
end for
$S'_w \leftarrow S_w \setminus BadStates$
if R'_w (which is R_w restricted to S'_w) is total tree relation then substitute w with $\langle S'_w, R'_w, I_w \cap S'_w\rangle$ in $\mathcal{M}$
else remove w from $\mathcal{M}$
return($\mathcal{M}$)
}

Example 2 *In Figure 2, the Kripke structure generated from the agents specifications, contains three worlds for the agents $protocol$, snd and rcv. The initial state is marked by 0 and the list of atomic propositions true at a state are written beside the circle representing that state. The values of symbols m_1, m_2, m_3 and m_4 are $inform(snd, rcv, p)$,*

$inform(snd, rcv, \neg p)$, $inform(rcv, snd, B_{rcv}p)$ *and* $inform(rcv, snd, B_{rcv}\neg p)$*, respectively. The first step of the synthesis is the creation of the accessibility relations. The agent* $protocol$ *has beliefs on the agent* snd *and the agent* rcv *thus, there are accessibility relations from its world to new created worlds for the two agents as believed by the agent* $protocol$*. These accessibility relations are illustrated by dotted edges. We have also accessibility relations shown by dashed edges from the world(s) representing the agent* snd *to new created world representing the agent* rcv *because the agent* snd *has beliefs on the agent* rcv*. In the second step, we have removed the states (with their edges) that are making conflicts (there are two states as colored in the world of the agent* snd*). Then, the resulting Kripke structure is a possible world representing the multi-agent system which can be used to check specified properties for the multi-agent system.*

5 BDI_{CTL} Model Checking

In this section, we present an extension of the standard CTL model checking algorithm [4]. Given a BDI_{CTL}-formula φ and a world of BDI_{CTL}-model $\mathcal{M}_w$ with a finite set of states (S_w), the model checking algorithm MAS-Sat(w, φ) (presented below) computes the set of states from the world w satisfying the BDI_{CTL} formula φ. This set is denoted $[\![\varphi]\!]_w$, and is computed in a recursive way, i.e. by computing for each sub-formula ψ of φ the set $[\![\psi]\!]_w$. In order to decide whether $s \models_{\mathcal{M}_w} \varphi$ we just have to check whether $s \in [\![\varphi]\!]_w$.

Algorithm 3 *MAS-Sat(w, φ)*

{
case φ *of*
$p \mid p \in AP$: $[\![\varphi]\!]_w \leftarrow \{s \mid p \in L(w,s)\}$
$O_j\psi \mid O_j\psi \in AP$: $[\![\varphi]\!]_w \leftarrow \{s \mid O_j\psi \in L(w,s)\}$
$O_j\psi \mid O_j\psi \notin AP$: *Let* v *be the world of the agent* j
and let wv *be the world of the agent* j
as viewed by the agent of the world w
$[\![\psi]\!]_{wv} \leftarrow$ *MAS-Sat(wv, ψ)*
$\mathcal{O}^{-1}_{(w,v)}([\![\psi]\!]_{wv}) \leftarrow \{\mathcal{A} \subseteq BDI_{(w,v)} \mid \mathcal{O}_{(w,v)}(\mathcal{A}) \subseteq [\![\psi]\!]_{wv}\}$
$[\![\varphi]\!]_w \leftarrow \{s \in S_w \mid TrueBDI_{(w,v)}(s) \subseteq \mathcal{O}^{-1}_{(w,v)}([\![\psi]\!]_{wv})\}$
$\neg\psi$: $[\![\varphi]\!]_w \leftarrow S_w \setminus$ *MAS-Sat(w, ψ)*
$\psi \vee \gamma$: $[\![\varphi]\!]_w \leftarrow$ *MAS-Sat(w, ψ)* $\cup$ *MAS-Sat(w, γ)*
$\exists X\psi$: $Q \leftarrow$ *MAS-Sat(w, ψ)*
$[\![\varphi]\!]_w \leftarrow \{s \in Q \mid \exists\langle s, s'\rangle \in R_w \wedge s' \in Q\}$
$\exists G\psi$: $[\![\varphi]\!]_w \leftarrow \nu Z.([\![\psi]\!]_w \cap \exists X\ Z)$
$\exists(\psi U\gamma)$: $[\![\varphi]\!]_w \leftarrow \mu Z.([\![\psi]\!]_w \cup ([\![\gamma]\!]_w \cap \exists X\ Z))$
end case
return($[\![\varphi]\!]_w$*)*
}

The standard model checking algorithm is adopted to accept formulas of the form $O_j\psi$ which are not BDI atoms. To compute the set of states satisfying these formulas, first we compute the satisfaction set $[\![\psi]\!]_{wv}$ of the sub-formula ψ, then we compute the set of BDI atoms ($\mathcal{O}^{-1}_{(w,v)}([\![\psi]\!]_{wv})$) whose accessible states are sub-sets of $[\![\psi]\!]_{wv}$. At the end, the satisfaction set of $O_j\psi$ is the states whose true BDI atoms are subsets of $\mathcal{O}^{-1}_{(w,v)}([\![\psi]\!]_{wv})$.

For the last two cases $\exists G\psi$ and $\exists(\psi U\gamma)$ we calculate a fix-point. The satisfaction set of $\exists G\psi$ is the greatest fix-point ($\nu Z.([\![\psi]\!]_w \cap \exists X\ Z)$), and the satisfaction set of $\exists(\psi U\gamma)$ is the least fix-point ($\mu Z.([\![\psi]\!]_w \cup ([\![\gamma]\!]_w \cap \exists X\ Z))$).

6 Conclusion and Related Work

We have presented a new approach to the verification of multi-agent systems, based on the use of possible worlds to describe the system, modal temporal logic to specify the properties, and a decision procedure based on model checking technique. One contribution is the presentation of an imperative multi-agent programming language, and a formal semantics for this language in terms of the BDI_{CTL} logic. The multi-agent program is used to systemically construct the agents state spaces. Our main contribution is the synthesis of these state spaces using the agents mental attitudes to generate the possible worlds structures. These possible worlds will be used by the adopted decision procedure to solve the problems of verification.

The notions of possible worlds is inspired by the works in [15, 17, 16] and the works in the field of multi-language systems [8, 9]. Other related work is in [1], where a finitely nested data structure is used to model the belief-desire-intention states. The authors of [11] present an automata theoretic approach to temporal modal logic restricted to the case of single nesting of beliefs, applied to the specification of knowledge-based systems.

In [22], the authors present the MABLE language for the specification of multi-agent systems. In this work, modalities are translated into nested data structures in the spirit of [1]. The author of [2] use a modified version of the AgentSpeak(L) language [14] to specify agents and to exploit existing model checkers. Both the works of [22] and [2] translate the specification into a SPIN specification to perform the verification. Effectively, the attitudes for the agents are reduced to predicates, and the verification involves only the temporal verification of those. In [13] a tool is provided to translate an interpreted system into SMV code, but the verification is limited to static epistemic properties, i.e. the temporal dimension is not present, and the approach is not fully symbolic. The work of [12] is concerned with verification of interpreted systems for model checking knowledge and time based on OBDD's.

Currently we are investigating the extension in many directions. One is the extension of the language to support the other types of expression. In particular the arithmetic expressions, by incorporating a tool for abstracting the program using the framework of predicate abstractions. Another problem which is taking our attention is the explosion problem, where techniques like the equivalence based reduction or space partition can be investigated. One of the

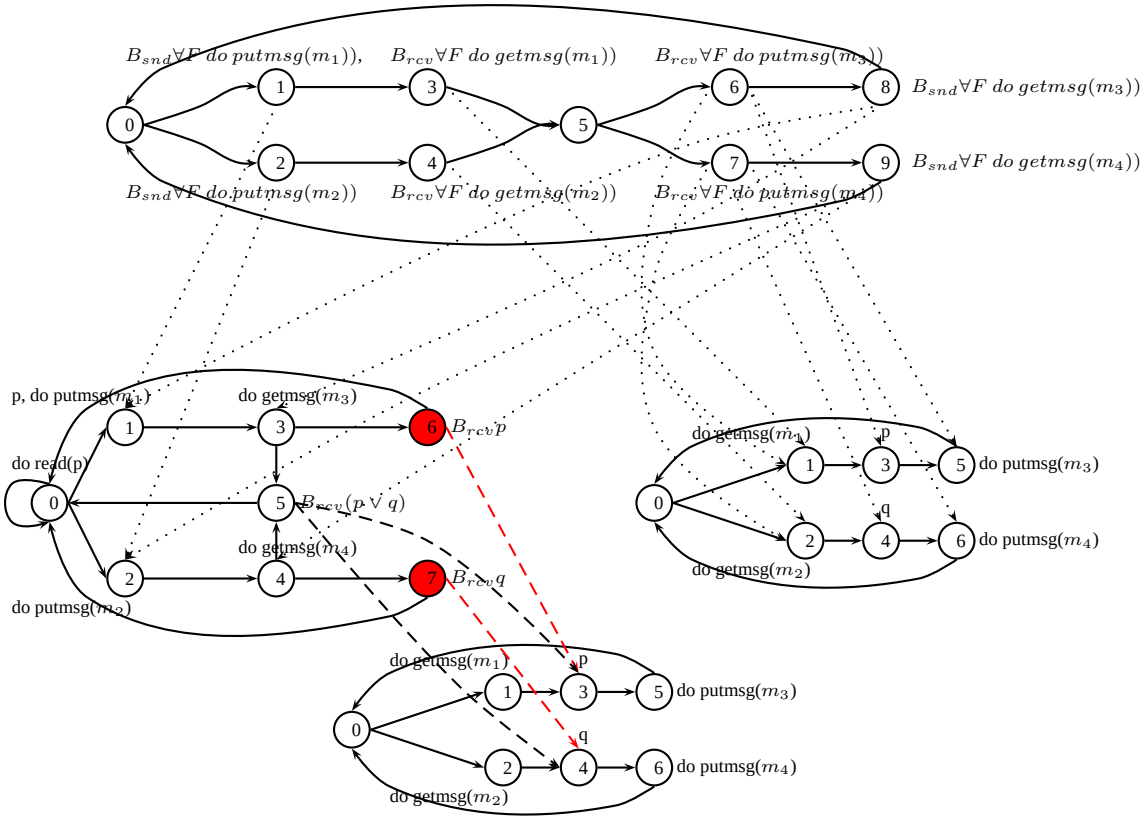


Figure 2: Construction of the possible worlds

most and interesting extension is to treat the case of functional dependencies between the mental attitudes, where a mental attitude is considered to be a function of one or more other mental attitudes.

References

[1] Benerecetti M., F. Giunchiglia, and L. Serafini (1998) Model checking multi-agent systems, *Journal of Logic and Computation* **8(3)**, pp. 401–423.

[2] Bordini R. H., M. Fisher, C. Pardavila, and M. Wooldridge (2003) Model checking AgentSpeak. *Proceedings of the Second International Joint Conference on Autonomous Agents and Multi-agent Systems (AAMAS'03)*.

[3] Bratman, M. E., D. Israel, and M. E. Pollack (1988) Plans and resource bounded practical reasoning, *Computational Intelligence* **4**, pp. 349–355.

[4] Clarke, E. M., O. Grumberg, and D. A. Peled (1999) *Model Checking*, MIT Press.

[5] Cohen P. R., and H. J. Levesque (1990) Intention is Choice with Commitment, *Artificial Intelligence* **42**, pp. 213–261.

[6] Doyle J. (1992) Rationality and its roles in reasoning, *Computational Intelligence* **8(2)**, pp. 376–409.

[7] Emerson E. A., and J. Srinivasan (1989) Branching time temporal logic, Linear Time, Branching Time and Partial Order, *Proceedings of Logics and Models for Concurrency*, Springer-Verlag, pp. 123–172.

[8] Ghidini C. and F. Giunchiglia (2001) Local Models Semantics, or Contextual Reasoning = Locality + Compatibility, *Artificial Intelligence* **127(2)**, pp. 221–259.

[9] Ghidini C. and L. Serafini (1994) Multi-language hierarchical logics (or: how we can do without modal logics), *Artificial Intelligence* **65**, pp. 29–70.

[10] Halpern J. Y., and Y. O. Moses (1990) A guide to completeness and complexity for modal logics of knowledge and belief, *Artificial Intelligence* **54**, pp. 319–379.

[11] van der Meyden R. and M. Y. Vardi (1998) Synthesis from Knowledge-Based Specifications, *Proceedings of the 9th International Conference on Concurrency Theory (CONCUR'98)*.

[12] Raimondi F. and A. Lomuscio (2004) Verification of multi-agent systems via ordered binary decision diagrams: an algorithm and its implementation, *Pro-*

ceedings of the First International Joint Conference on Autonomous Agents and Multi-agent Systems (AAMAS'04).

[13] Raimondi F. and A. Lomuscio (2003) A tool for specification and verification of epistemic and temporal properties of multi-agent system, *Electronic Notes in Theoretical Computer Science*.

[14] Rao A. S. (1996) AgentSpeak(L): BDI agents speak out in a logical computable language, *Lecture Notes in Computer Science*.

[15] Rao A. S., and M. Georgeff (1998) Decision procedures for BDI logics, *Journal of Logic and Computation* **8(3)**, pp. 293–344.

[16] Rao A. S., and M. P. Georgeff (1991) Modeling rational agents within a BDI architecture, *Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning*, Morgan Kaufmann.

[17] Rao A. S., and M. P. Georgeff (1992) An abstract architecture for rational agents, *Knowledge Representation and Reasoning*, pp. 439–449.

[18] Rosenschein S. J., and L. P. Kaelbling (1986) The synthesis of digital machines with provable epistemic properties, *Proceedings of the First Conference on Theoretical Aspects of Reasoning about Knowledge*, Morgan Kaufmann, 1986.

[19] Shoham Y. (1991) Agent0: A simple agent language and its interpreter, *Proceedings of the Ninth National Conference on Artificial Intelligence (AAAI91)*, pp. 704–709.

[20] Woodridge M. (2000) Computationally grounded theories of agency, *Fourth International Conference on Multi-Agent Systems (ICMAS-2000)*, pp. 13–20.

[21] Woodridge M., and M. Fisher (1994) A decision procedure for a temporal belief logic, *Proceedings of the First International Conference on Temporal Logic*.

[22] Wooldridge M, Fisher M., M.P. Huget, and S. Parsons (2002) Model checking multi-agent systems with MABLE. *Proceedings of the First International Joint Conference on Autonomous Agents and Multi-agent Systems (AAMAS'02)*.

Improving Branch Prediction Performance with A Generalized Design for Dynamic Branch Predictors

Wei-Ming Lin, Ramu Madhavaram and An-Yi Yang
Department of Electrical Engineering
University of Texas at San Antonio
San Antonio, TX 78249-0669, USA
WeiMing.Lin@utsa.edu

Keywords: Branch Prediction, Two-Level Predictor, Gshare, Generalized Branch Predictor

Received: January 1, 2004

Pipeling delays from conditional branches are major obstacles to achieving a high performance CPU. Precise branch prediction is required to overcome this performance limitation imposed on high performance architecture and is the key to many techniques for enhancing and exploiting Instruction-Level Parallelism (ILP). A generalized branch predictor is proposed in this paper. This predictor is a general case of most of the predictors used nowadays, including One-Level Predictor, Two-level predictor, Gshare, and all their close and distant variations. Exact pros and cons of different predictors are clearly analyzed under the same general format. The concept in the traditional Gshare predictor is then extended to form a more flexible predictor under the same construct. By following this generalized design scheme, we are able to fine-tune various composing parameters to reach an optimal predictor and even allow the predictor to adjust according to various types of applications. From our simulation results, it is evident that significant improvement over traditional predictors is achieved without incurring any additional hardware.

Povzetek: Članek obravnava delovanje CPU in napovedovanje razmejitev.

1 Introduction

In the past decade, by taking advantage of RISC architecture and advanced VLSI technology, computer designers were able to exploit more Instruction-Level Parallelism (ILP) by using deeper pipelines, wider issue rates and superscalar techniques. However, these techniques suffer from disruption caused by branches during the issue of instructions to functional units. How to appease such a performance-degrading effect from branch instructions, which typically make up twenty or more percentage of an instruction stream, has to be paid with more attention.

Branch prediction is a common technique used to overcome this performance limitation imposed on high performance architectures and is the key to many techniques for enhancing ILP. Branch prediction essentially involves a guess on the likely stream direction that is to take place after a branch instruction; whenever such a guess is correct, penalty in pipeline delay is either reduced or completely avoided. There have been various branch prediction schemes proposed in this area [1, 2, 6, 7, 8, 10, 13, 15, 21]. They are usually classified as static or dynamic according to how prediction is made. Static prediction schemes always assume same outcome for any given branch, whereas a dynamic scheme uses run-time behavior of branches to adjust the database for later predictions. Focus of this paper is on the dynamic ones which usually show far better prediction accuracy than the static ones.

A typical dynamic prediction mechanism relies on a prediction table, or the so-called Pattern History Table (PHT) to record the behavior of past branches. One of the very early predictors used was the One-Level Predictor which has a one-dimensional PHT and uses only program counter (PC) as the index to retrieve past branch behavior and record new branch behavior. Usually one-bit or two-bit saturating up-down counters are used in the PHT to record the behavior of branches. When these branches are encountered again they are predicted based on their entries in the table, i.e., based on their previous behavior. It was followed by the more widely used Two-Level Adaptive Predictor [17], which has the PHT but organized into a two-dimensional table. Such a PHT is addressed using both the PC index and a history register (HR) index. The HR is used to record behaviors of either the most recent per-address branch or the most recent global branches. The two-level predictor easily outperforms the one-level one by exploiting potential correlation between branches at run-time.

Another very popular predictor design, Gshare, was proposed in [9]. Unlike the previously proposed designs, Gshare addresses the PHT with a blended index between global history and the PC. Based on Gshare, some other designs including LGshare [4] have also been proposed. Although these Gshare-based designs usually yields better prediction results than the traditional two-level predictors in most cases, the exact reason behind their design and their relationship with the two-level predictors has never

been discussed, nor has a complete analysis on their benefits ever been presented.

After carefully analyzing the structure of all the aforementioned predictors, we propose a general prediction scheme which encompasses all these popular designs. This generalized scheme displays a standard organization for the PHT and a flexible selection for HR index. This proposed scheme provides a platform with which one can easily understand the similarities and/or differences among different predictors proposed so far. In addition, based on this, we can provide a more systematic explanation for the benefits a predictor provides and the drawbacks it may come with.

The remainder of the paper is organized as follows. A brief overview of the well-known counter-based dynamic branch prediction schemes is presented in section 2. The proposed generalized predictor is then described in the following section along with its variations. In section 4, our simulation and performance comparison results are presented. Concluding remarks are given in the last section.

2 Dynamic Branch Prediction

There have been many dynamic branch prediction schemes proposed in the past decade. A few representative ones are described in the following for the sake of completeness.

2.1 One-Level Predictor

The prediction table is usually indexed by the lower-order address bits in the program counter (PC), although other portions of the PC have been used as well. Figure 1 illustrates the design of such a scheme. Each entry in the predic-

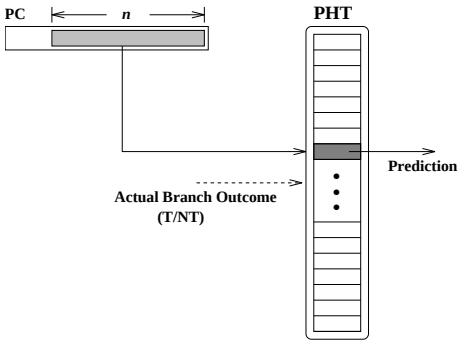


Figure 1: Implementation of Simple One-Level Branch Prediction

tion table (PHT) is used to provide prediction information for the branch instruction mapped to it, and is implemented by a counter which goes up or down according to the actual outcome of the corresponding branch instruction. Each branch is predicted based on its most recent outcome. Instead of the simple one-bit counter, a well-known two-bit up-down counter has been extensively used in this scheme so as to render a damping effect which enhances prediction accuracy for typical reentrant loop constructs. Damage caused by alternating occurrences between two aliasing branches can also be alleviated using the two-bit counters. Such an observation prompts most later advanced designs to use such a two-bit counter prediction table as a design base.

2.2 Correlation-based or Two-Level Adaptive Predictor

Outcome of a branch is usually affected by some previously executed branches. Such a correlation could exist among different branch instructions executed temporally close to one and other, or simply refers to the effect on a branch from its own recent execution behavior. The latter one has been partially considered in the simple one-level two-bit counter design. Such an approach requires a separate table, the so-called history table, to record the necessary history information. A general design block diagram is shown in Figure 2 in which the PHT organized as a two-dimensional table is addressed by two separate indices, the PC index and the history index. History information established in a history table can be either in per-address (per-branch) format as shown in Figure 2 or in global format as shown in Figure 3. In a per-address case, a

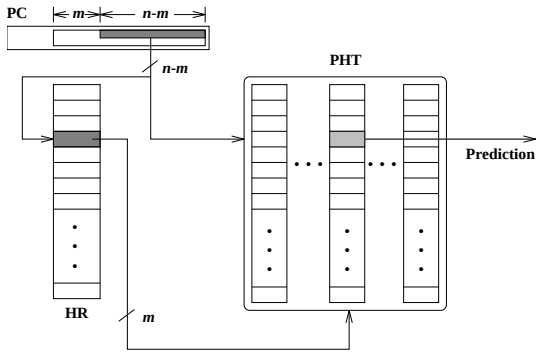


Figure 2: Implementation of Per-Address Correlation-based Branch Prediction

per-address history table is needed which also is addressed by the PC index. A shift register, so-called History Register (HR), is usually used to implement each such entry. On the other hand, for the global format, only one HR is needed, as shown in Figure 3. This aims at exploiting the correlation in behavior existing in most programs between recurring identical branches (as in the per-address case) or between distinct branches adjacent in time (as in the global case). HR index and PC index combined are then used to locate the counter in the PHT for prediction. It is shown that [19] global history schemes perform well with integer programs while per-address history schemes are better for floating point programs. Also, note that the PC index for history table does not have to come from the same least significant portion of PC that the PC index for PHT normally uses. The so-called "per-address" refers to the one that uses the least significant bits of PC for such an index, while the "per-set" refers to the selections otherwise. In general, such

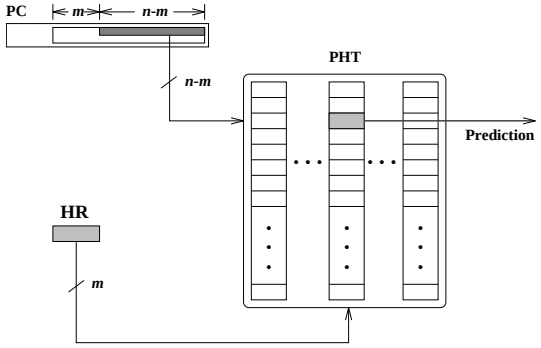


Figure 3: Implementation of Gloabl Correlation-based Branch Prediction

a selection does not lead to any significant discrepancy in performance.

2.3 Gshare Predictor

In the Gshare scheme [9], as shown in Figure 4, the prediction table is addressed by an index established by XORing a global history and part of the PC index. Gshare scheme does lead to improvement in most cases compared to a simple two-level predictor; however, the exact cause for such an improvement has never been clearly analyzed. (Note that, in one of the original Gshare designs, the XORing function is performed over the entire PC index; that is, m is set to be equal to n, which in general leads to worse performance than a simple two-level predictor.)

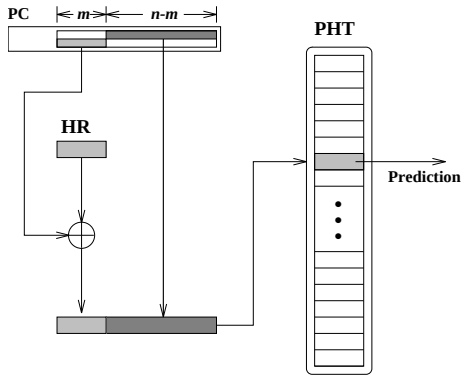


Figure 4: Implementation of Sharing Index Branch (*Gshare*) Prediction

2.4 Others Well-Known Predictors

The possibility of combining different branch predictors is exploited by McFarling in [9]. It comes from the observation that some schemes work well on one type of programs while not so on another. The selective scheme is implemented with two different predictors, with each making prediction separately. A third table is then used to make decision between the two prediction outcomes based on various program scenarios. Such a scheme is claimed to perform well on different circumstances, yet it has a hardware cost roughly three times of what a non-selective one would cost. A predictor called LGshare has also been proposed [4] to further improve on Gshare by using both global as well as per-address history of a branch to predict its behavior. Among many more others in this field, a new predictor discussed in [6] is based on Simultaneous Subordinate MicroThreading (SSMT), which provides a new means to improve branch prediction accuracy. SSMT machines run multiple concurrent microthreads in support of the primary thread to dynamically construct microthreads that can speculatively and accurately pre-compute branch outcomes along frequently mispredicted paths. Another technique is introduced in [5] to reduce the pattern history table interference by dynamically identifying some easily predictable branches and inhibiting the pattern history table update for these branches.

In general, there are a few types of well-known potential problems that would lead to a misprediction result due to the nature of the predictor employed:

- Initialization –
  Every branch instruction that has a predictable behavior needs to have its behavior history properly established in the prediction table before a meaningful prediction can be made.

- Alias –
  This problem occurs when different branches are mapped to the same entry in the PHT. Such a problem is unavoidable unless a sufficiently large number of entries to cover all potential program sizes are provided.

- Undetected Correlation –
  Due to the limited size of history register, correlation among branches far apart in time/trace may not be detected.

- "Random" (Unpredictable) Branch Behavior –
  A branch's behavior, either at times or throughout the life of the program, may be simply run-time data-dependent which is either completely "random" or unpredictable based on any of the known branch prediction schemes.

Some of the above problems may further intertwine with each other. For example, if the overall size of the predictor table is to remain the same, by increasing the history depth (the history register size) to allow more potential correlation to be detected, alias problem between different branches would worsen. It is part of our goals in our proposed generalized predictor to determine the pros and cons among the various predictors and to incorporate an additional flexibility in our predictor to accommodate for various programs that may call for different prediction techniques.

3 Proposed Generalized Predictor

A generalized predictor is proposed in this section to show that most of the predictors mentioned above fall under this category. With the introduction of this design scheme, potential performance-influencing factors can be more clearly analyzed. Additional design flexibility is also incorporated in this design to allow adjustment of certain design parameters to accommodate for various program behaviors. In order to have a fair comparison among all predictors, all are assumed to be of unified cost, i.e. predictors with similar hardware are compared.

3.1 Generalized Predictor

Figure 5 illustrates the proposed generalized predictor design. In this design, the PHT is organized as a two-

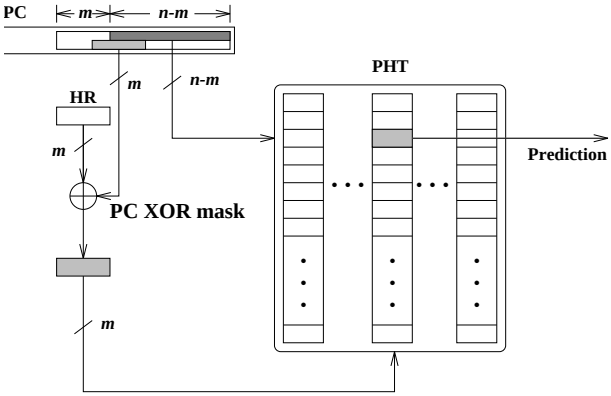


Figure 5: The Proposed Generalized Predictor

dimensional table similar to the traditional two-level predictor. The primary hardware cost resides in the PHT, the size of which is thus fixed at 2^n. The lower portion of PC from bit 0 to bit $n-m-1$ is used to address the PHT as the row index, while the column index is composed by XORing the m-bit HR and a "floating" portion of PC. This portion of PC is called as the "PC XOR mask" throughout this paper.

3.2 Special Case #1: One-Level Predictor

The one-level predictor as shown in Figure 1 can be reorganized as a special case of the proposed generalized predictor. Figure 6 illustrates such an arrangement. By having the PC XOR mask fixed at the highest position of the n-bit index, and XOR-ing it with the non-existing HR (0's throughout the HR content), the original one-dimensional PHT is then re-arranged as a two-dimensional PHT. The sequence of addressing in the original one-dimensional PHT is then mapped to a column-major-order sequence in this new two-dimensional PHT, i.e. one column followed by the next one. Such a rearrangement presents us a platform for a direct comparison between any advancement from this technique and the original one.

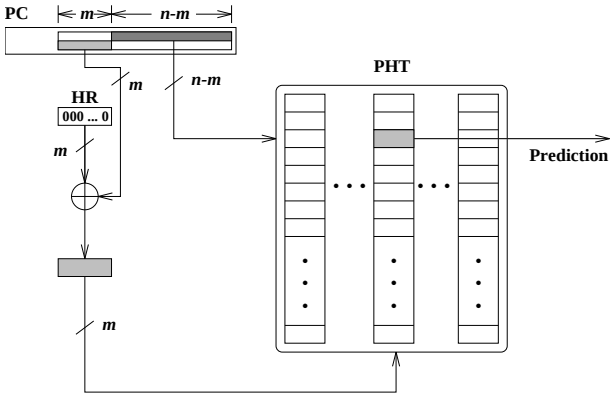


Figure 6: One-Level Predictor as A Special Case

3.3 Special Case #2: Two-Level Predictor

Comparing the two-level predictor with the one-level Predictor, one can obviously see that the former intends to improve prediction accuracy by using more history to exploit correlation information among different branches' behavior. However, with the cost fixed, the two-level predictor introduces potentially more alias problem into the picture by having a smaller PC index (row index) used. That is, for every additional bit of HR employed (m being increased by 1), the number of row entries of the PHT is reduced by half. It has been shown that such a tradeoff usually is worthwhile to a certain extent of m due to the following reasons:

- Benefit from exploiting correlation among branches usually outweighs the potential performance loss from the alias problems thus incurred.
- Alias problem between two branches thus incurred can sometimes be relieved if they have a different global history pattern in HR even though they are "aliased" into the same row entry. In this case, the alias problem is removed since their prediction entries are mapped into different columns albeit in the same row.

The two-level predictor based on a global history HR as shown in Figure 3 can be also reorganized as a special case of the proposed generalized predictor. Figure 7 illustrates such an arrangement. The new arrangement has the XOR mask confined to within the row index, i.e. the first $n-m$ bits of PC. This results in no change of prediction accuracy because the mapping of branches is merely swapped around among the columns i.e. branches mapped to one column are now mapped to a different column and this takes place symmetrically for all the branches. This comes from a simple understanding of how XOR function applies to a given bit pattern. For example, in Figure 8, two-bit column indices from a two-bit HR are one-to-one mapped to different set of indices when XORed with a different XOR-mask values from PC. Obviously, if this mask portion of PC is within the row index portion of PC as indicated in this special case for two-level predictor, then each branch

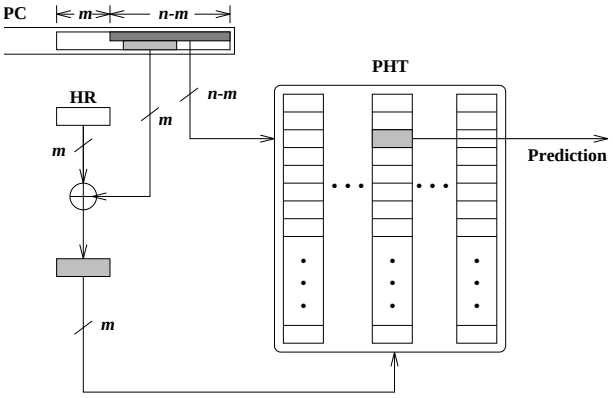


Figure 7: Two-Level Predictor as A Special Case

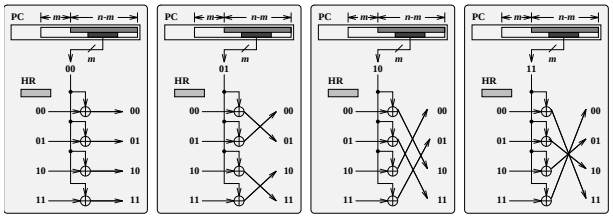


Figure 8: Index Swapping Effect from XOR Function

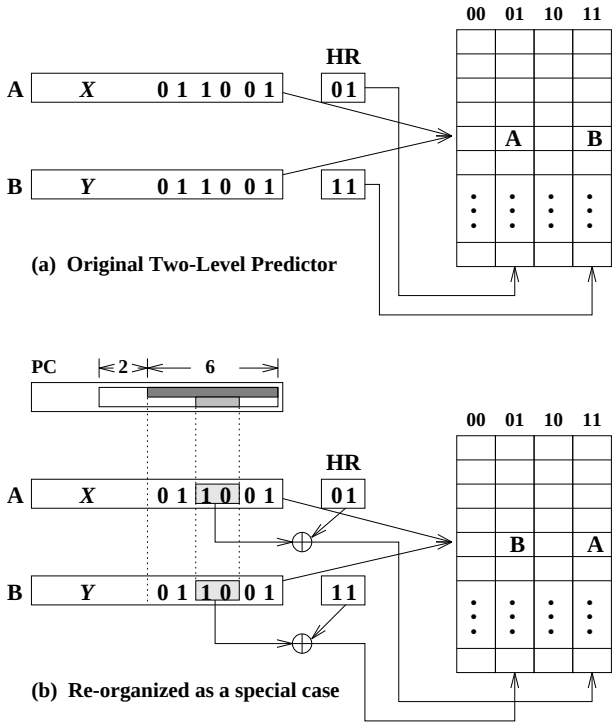


Figure 9: Index Swapping between Two Aliased Instructions

will be still mapped to the same row except that it may be using a different column index mapping according to its run-time HR content. Thus, the prediction result would remain the same comparing the original arrangement and the new arrangement pertaining to a non-alias case. For an alias case between two different branches that are mapped to the same row, the alias effect still remains the same since both instructions would have the same XOR-mask content from within their identical row index content. An example showing such a swapping between two instructions is given in Figure 9. It happens so because no extra PC information is used and thus two aliased branches cannot be differentiated with the same PC index. As shown in Figure 9 two aliased branches A and B with different histories are mapped to different columns along the same row before XORing with XOR-mask from the PC. Consequently, after XORing as shown in the second figure, both A and B are mapped to different columns but are just swapped around which does not lead to different prediction result from the two-level predictor. So it can be concluded that two-level predictor is also a special case of this generalized predictor.

3.4 Special Case #3: Gshare

Gshare is one of the global two-level predictors, which exploits correlation by basing the prediction on the outcome of the recently executed branches. The XORing is done to incorporate history information into the PC index thereby differentiating interfering branches with the help of history bits. Similarly Gshare can be organized as a special case of our proposed predictor, through which we can easily analyze the benefits brought by this technique. Figure 10 demonstrates this new arrangement. In the following, a

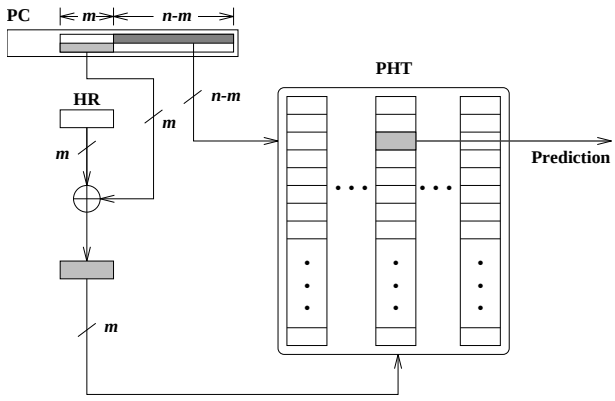


Figure 10: Gshare Predictor as A Special Case

thorough analysis on how Gshare compares to the two special cases thus far introduced is given.

3.4.1 Gshare vs. One-Level Predictor

As shown earlier, the original Gshare predictor is similar to one-level predictor in the way its PHT is organized with the difference that Gshare additionally uses history information and so produces much better prediction accuracies than the one-level predictor. Similar to two-level predictor, Gshare benefits from exploiting correlation information from the use of HR, but it addresses the tradeoff between

alias problem and loss of correlation in a more delicate manner, which is to be described in the following section.

3.4.2 Gshare vs. Two-Level Predictor

Gshare, when compared to the two-level predictor, is said to be advantageous due to the XORing effect and can be explained as shown in Figure 11. The benefit comes from that fact that, in most programs, global history (HR) content tends to exhibit one dominant pattern, either $00\ldots00$ or $11\ldots11$, due to loop constructs especially when the history depth used (m) is small. This claim has also been confirmed by our simulation results. Consider four aliased branches A, B, C and D that are aliased into the same row in the PHT. The mapping of these branches with different values of HR is shown for both two-level predictor in (a) and Gshare predictor in (b). A_{00}, B_{00}, C_{00} and D_{00}, where the subscripts denote the HR contents, are mapped to different columns of the same row in Gshare as compared to same column in two-level. Same scenario applies to the case when HR has a content of 11. This is one of the advantages of Gshare because aliased branches with most dominant history patterns are now mapped to different locations thereby reducing destructive overlapping. That is, assuming that they have the same history, all the four branches are mapped to the same column in a two-level predictor and so cannot be distinguished. In Gshare on the other hand, these aliased branches are "dispersed" to different columns and so the problem is resolved.

The mapping difference described above is the only distinction between the Gshare and the two-level predictor, and such a distinction may not always favor the Gshare due to different program behavior. A very important conclusion that can be drawn from this is that the Gshare is essentially identical to the two-level predictor if the $(n-m)$-bit row index does not lead to any alias problem. That is, the dispersion of dominating entries among aliasing branches no long exists, thus leading to no more difference in prediction result.

Size of the XOR mask also plays a very important role in Gshare's potential performance. Similar to the two-level predictor, the larger the mask is (larger m), the more history correlation among branches can be exploited. On the other hand, a larger mask leads to more alias problems, although the column mapping of Gshare may provide a better alias-differentiating support than the two-level one from our discovery. As one of the most extreme case, in one of the original Gshare designs, the XORing function is performed over the entire PC index; that is, m is set to be equal to n. This in general leads to an undesirable performance due to its excessive alias problems.

3.5 The Proposed Generalized Predictor with Extensions

A generalized predictor as proposed can be specified with the position of the m-bit XOR mask in PC. Let the least significant bit of this mask be starting at bit W_s; that is, the mask ranges from bit W_s to $W_s + m - 1$. Each of the special cases is then defined as in the following:

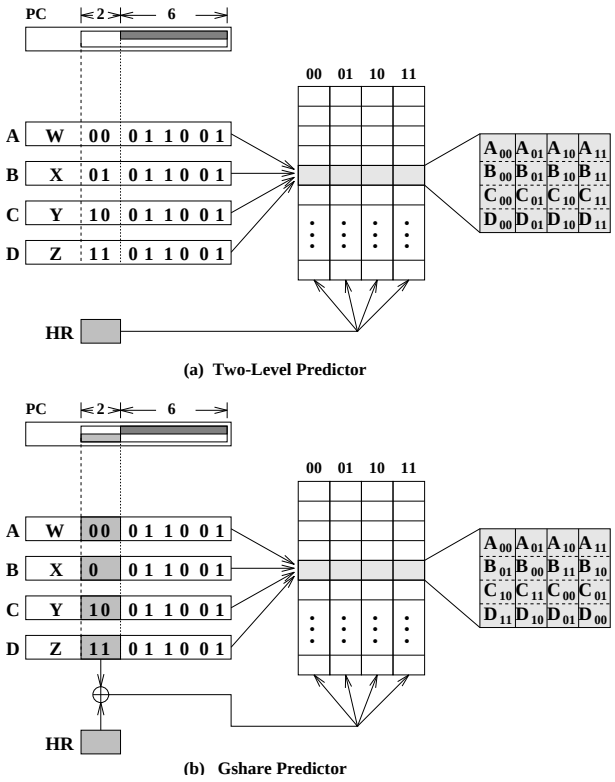


Figure 11: Difference between Gshare Predictor and Two-Level Predictor

Case	Range of W_s	HR value
(a)	$0 \le W_s \le n-2m$	run-time
(b)	$n-2m+1 \le W_s \le n-m-1$	run-time
(c)	$W_s = n-m$	$00\ldots00$
(d)	$W_s = n-m$	run-time
(e)	$n-m+1 \le W_s$	run-time

Clearly, case (a) corresponds to the two-level predictor, case (c) to the one-level predictor and case (d) corresponds to the Gshare predictor, while cases (b) and (e) have never been addressed.

Case (b) is essentially a combination of two-level and Gshare predictors. Such a hybrid design displays a various degree of dispersion on dominating entries of aliasing branches. Figure 12 shows an example similar to the one presented in Figure 11. As a compromising point in between the two-level one that shows zero dispersing effect and the Gshare that has a 100% dispersing effect, this special case exhibits a 50% dispersing capability. Branch A

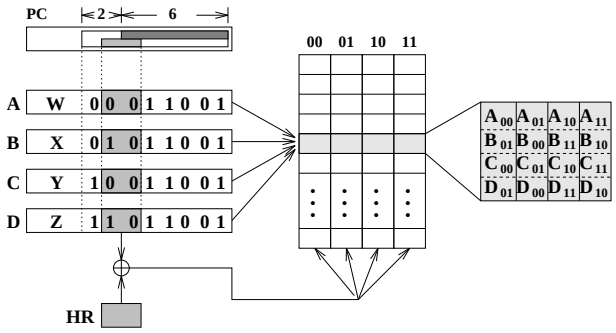


Figure 12: A Hybrid Design of Two-Level and Gshare Predictors

and C remain aligned and so do B and D among the four aliasing branches. Performance from such a hybrid predictor is usually unpredictable due to its nature.

Case (e) is an extension of Gshare aimed at programs with larger address space and/or with a small PHT. The example in Figure 13 shows that, between the two aliasing instructions A and B, the dispersing effect does not take place in the traditional Gshare since both instructions have the same XOR mask value. That is, under such a circumstance, Gshare performs exactly like the two-level predictor. Moving the XOR mask to the higher portion of PC allows the dispersion effect to re-emerge.

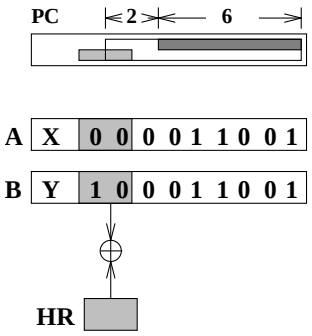


Figure 13: An Extended Design from Gshare

4 Simulation

Our trace analysis and simulation are performed on a SPARC 20 system. Data are obtained using Shade version 5.25 analyzing program. Shade is a dynamic code tracer, which combines instruction set simulations, trace generation and custom trace analysis in a process. Our test programs include a benchmark program from the Stanford Body Benchmark Suite, sfloat, and seven standard Unix utility programs. A brief description of these programs on various aspects of their branch instructions is given in Table 1. A trace analysis has been performed on these eight test programs to show the percentage improvement in misprediction rate.

program	# instr	# taken	# not-taken
sfloat	7730261	111170	125382
ls	1207004	112180	91466
gcc	677017	61258	58096
cc	543900	50519	46536
chmod	492158	45812	42507
grep	491016	45672	42382
awk	490911	45524	42365
pack	486776	45159	42173

Table 1: Description of Test Programs

4.1 Simulation Results

A series of simulation runs are performed by varying the following three parameters on one-bit and two-bit prediction schemes: (1) Number of index bits (n), (2) Number of history bits (m), and (3) Shift in the window (W_s). Performance comparison results are plotted after taking the average of improvement in miss prediction rate for all the above eight programs. The "miss rate improvement percentage" is defined as:

$$\frac{M_{\text{two-level}} - M_{\text{generalized}}}{M_{\text{two-level}}} \times 100$$

where $M_{\text{two-level}}$ and $M_{\text{generalized}}$ denote the miss rate of two-level prediction scheme and that of the generalized one, respectively. Each point in these results corresponds to the average of results from the eight test programs. Figure 14, Figure 15 and Figure 16 show the results for both one-bit and two-bit counters prediction schemes.

We can see that, from all these figures, performance of the generalized predictor is identical to that of the two-level one (i.e. improvement equals to 0) when

$$0 \leq W_s \leq n - 2m$$

Gshare's results (when $W_s = n-m$), in general, are among the better ones for $n = 11$, but are outperformed by most of cases with larger W_s values (the Extended Gshare scheme) for $n = 10$ and $n = 9$. This finding verifies our analysis that the Extended Gshare scheme allows the dispersion effect to reappear when a small PHT is used. The Hybrid scheme (when $n - 2m + 1 \leq W_s \leq n - m - 1$) does not distinguish itself clearly from the two-level one.

5 Conclusion

In this paper, we propose a generalized branch predictor and show that most of the commonly used predictors are actually special cases of this generalized predictor. Similarities and differences among predictors are clearly identified. Based on this construct, we are able to easily analyze and compare the benefits and drawbacks among different predictor designs. We also show that a simple extension of

the Gshare design, a direct notion from the generalized design, can outperform Gshare in many cases. This is an improvement at no additional cost on hardware. A dynamic selection of the XOR-mask position according to the nature of the program may bring additional improvement. Also, one potential direction that is worthwhile looking into is the understanding and classification of different kinds of conditional branches, which may help predict the otherwise declared "random" branches that have not been addressed by the prediction methods investigated so far.

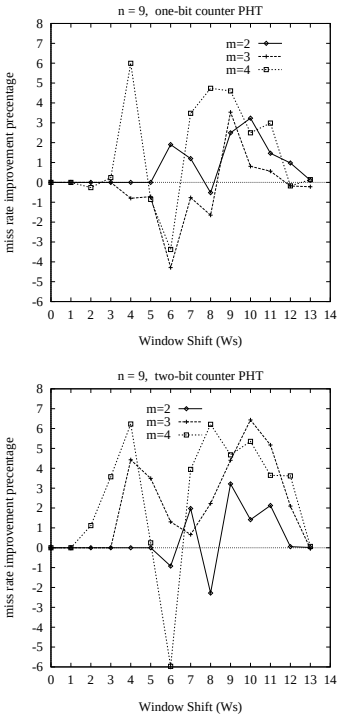


Figure 14: Improvement Results versus Window Shift for $n = 9$

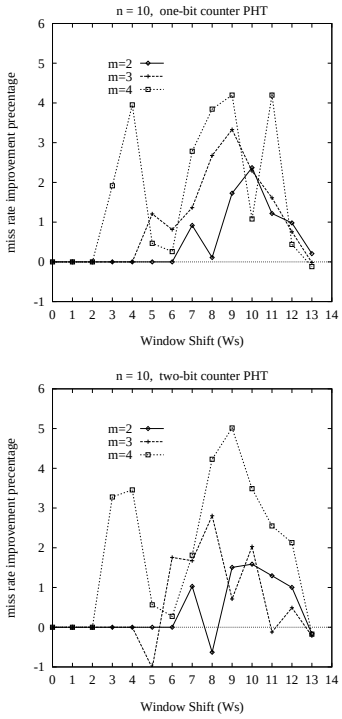


Figure 15: Improvement Results versus Window Shift for $n = 10$

References

[1] T. Ball and J. Larus, "Branch Prediction for Free," *Proc. ACM SIGPLAN 1993 conf. on Prog. Lang. Design and Implementation*, June, 1993.

[2] B. Bray and M. J. Flynn, "Strategies for Branch Target Buffers," *24th Workshop on Microprogramming and Microarchitecture*, 1991, p.42-p.49.

[3] B. Calder and D. Grunwald, "Fast & Accurate Instruction Fetch and Branch Prediction," *Intl. Symp. on Computer Architecture*, April, 1994.

[4] M.-C. Chang and Y.-W. CHou, "Branch Prediction using both Global and Local Branch History information," *Computers and Digital Techniques, IEE Proceedings, Volume: 149 Issue: 2*, March 2002.

[5] P.-Y. Chang, M. Evers and Y.N. Patt, "Improving branch prediction accuracy by reducing pattern history table interference," *Parallel Architectures and Compilation Techniques*, 1996.

[6] R. S. Chappell, F. Tseng, A. Yaoz and Y. N. Patt, "Difficult-path Branch Prediction Using Subordinate Microthreads," *Proc. 29th Annual International Symposium on Computer Architecture*, 2002.

[7] J. Fisher and S. Freudenberger, "Predicting Conditional Branch Direction From Previous Runs of a Program," *Proc. 5th Annual Intl. Conf. on Architectural Support for Prog. Lang. and Operating System*, October, 1992.

[8] J. K. F. Lee and A. Smith, "Branch Prediction Strategies and Branch Target Buffer Design," *IEEE Computer*, January, 1984, p.6-p.22.

[9] S. McFarling, "Combining Branch Predictor," *Technical Report, Digital Western Research Laboratory*, June, 1993.

[10] S. McFarling and J. Hennessy, "Reducing the Cost of Branches," *The 13th Annual Intl. Symposium of Computer Architecture*, 1986, p.396-p.403.

[11] S. Pan, K. So, and J. Rahmeh, "Improving the Accuracy of Dynamic Branch Prediction Using Branch Correlation," *Proc. 5th Annual Intl. Conf. on Architectural Support for Prog. Lang. and Operating System*, Oct. 1992.

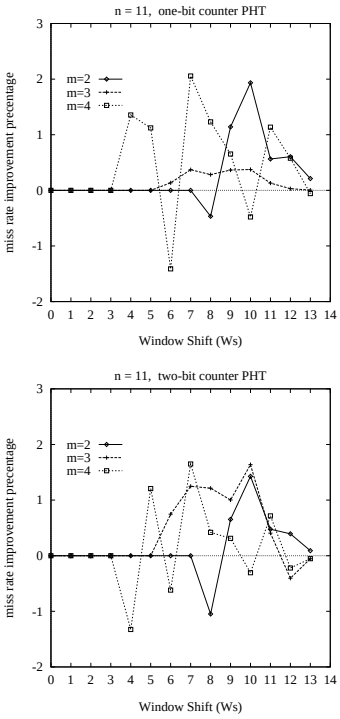


Figure 16: Improvement Results versus Window Shift for $n = 11$

[12] D. Patterson and J. Hennessy, "Computer Architecture: A Quantitative Approach, 2nd Edition," *Morgan Kaufmann Publishers, Inc.*, 1995.

[13] C. Perleberg and A. J. Smith, "Branch Target Buffer Design and Optimization," *IEEE Transactions on Computers*, April, 1993, P396-412.

[14] J. Smith, "A Study of Branch Prediction Strategies," *Proc. 8th Annual Intl. Symp. on Computer Architecture*, May, 1981, p.135-p.147.

[15] Z. Su and M. Zhou, "A Comparative Analysis of Branch Prediction Schemes," *Technical Report, University of California at Berkeley*, 1995.

[16] "Shade Manual," *Sun Microsystems*, 1995.

[17] T. Yeh and Y. Patt, "Two-level Adaptive Branch Prediction," *Proc. 24th Annual ACM/IEEE Intl. Symp. and Workshop on Microarchitecture*, Nov. 1991.

[18] T. Yeh and Y. Patt, "Alternative Implementations of Two-level Adaptive Branch Prediction," *Proc. 19th International Symp. on Computer Architecture*, May. 1992.

[19] T. Yeh and Y. Patt, "A Comparison of Dynamic Branch Predictors that use Two Levels of Branch History," *Proc. 20th Annual Intl. Symp. on Computer Architecture*, May. 1993.

[20] T. Yeh and Y. Patt, "Two-level Adaptive Branch Prediction and Instruction Fetch Mechanism for High Performance Superscalar Processors," *Computer Science and Engineering Div. Tech. Report CSE-TR-182-93, University of Michigan*, Oct. 1993.

[21] C. Young and M. Smith, "Improving the Accuracy of Static Branch Prediction Using Branch Correlation," *Technical Report 06-95, Center for Research in Computing Technology, Harvard University*, March, 1995.

[22] C. Young, N. Gloy and M. Smith, "A Comparative Analysis of schemes for Correlated branch Prediction," *Proc. 22nd Annual Intl. Symp. on Computer Architecture*, June, 1995.

Construction of Patient Specific Virtual Models of Medical Phenomena

Božidar Potočnik, Dušan Heric, Damjan Zazula, Boris Cigale and Daniel Bernad
University of Maribor, Faculty of Electrical Engineering and Computer Science
Smetanova 17, 2000 Maribor, Slovenia
E-mail: bozo.potocnik@uni-mb.si

Tomaž Tomažič
Teaching Hospital of Maribor
Ljubljanska 5, 2000 Maribor, Slovenia

Keywords: virtual medical model, image processing, modelling, simulation

Received: February 10, 2004

A framework for construction of virtual models of the medical phenomena is proposed. Major construction steps are discussed in detail. The construction of virtual medical model is guided from acquisition of patient imaging material, to 3D reconstruction based on the image processing, to the basic modelling and simulation approaches. This framework is demonstrated on human knee joint virtual model construction. Statistical assessment of the built knee joint model points out sufficient quality and accuracy. Model assessment from clinical point of view confirmed this evaluation, and, simultaneously, verified the proposed construction chain as very prospective.

Povzetek: Razvita je metoda za prikazovanje medicinskih pojavov na pacientovem kolenu.

1 Introduction

Atlases and 3D human organ models for "typical" patient do not suffice in a modern medical practice anymore. Successful diagnosing and decision-making in medicine today is unavoidably dependent on relevant patient specific information. Such information is mainly extracted by using non-invasive methods like medical imaging techniques, e.g. ultrasonography and magnetic resonance (MR) imaging. Furthermore, a need to adapt atlases and organ models for specific patient anatomy emerged. This need is intensified when planning surgeries. Surgeons namely face a problem when imagining a detailed surgery in advance, although having MRI recordings at their disposal, for example. They lack a model that would offer a virtual walk through the tissues and organs, and maybe an option of virtual testing of some specific surgery detail. A desired solution, of course, must incorporate a thorough and reliable computer support, which is not available in today's medical devices and computer software.

The models of organs, appropriate for surgical planning, should be available in their close-to-natural constellation either for individual usage, i.e. each organ separated from the others, or for grouping them together in arbitrary combinations. The obtained computer models should be aimed at any virtual (spatial) inspection and scanning along arbitrary cross-sections in all directions. The most desired option is a kind of virtual travel through models, possibly equipped with a collision detection module. The most challenging feature of such models is, however, virtual surgery.

A generic procedure for constructing virtual medical models is presented in this paper. After a survey of related work in Section 2, major construction steps are outlined on an example of human knee joint in Section 3. Section 4 presents simulation results and quantitative assessment of quality for constructed models, followed by a discussion section which emphasises current modelling approaches and potential difficulties with the process of construction. Our conclusions also stress the applicative value of such models in the clinical practice. The present work summarizes a part of the SimBio project results [15].

2 Related work

Current state of the art in the computer science and medical devices already enable individualisation of patient data processing. Various approaches to biomedical image processing, object recognition, and reconstruction have been developed [2, 5, 6, 12]. None of those methods cited is, however, general and equally applicable in different situations. On the other hand, medical imaging devices are daily used for patient diagnosis. Such examinations are relatively low-cost. The technology thus assures all possibilities for construction of virtual medical models.

There already exist few solutions for the medical field, where majority deal with an organ reconstruction/visualisation and virtual inspection. Reference [17] brings comprehensive review on this topic, while [7] describes virtual endoscopy as an example of virtual organ inspection. Some models tackled a virtual surgery, especially a surgical planning [9, 20]. There exists also other appli-

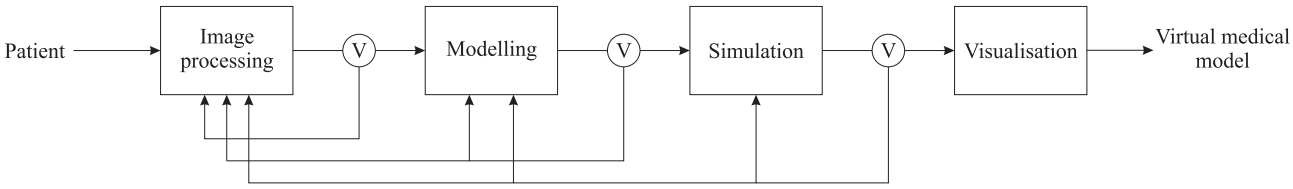


Figure 1: Major steps for construction of virtual medical model. Verification procedure (encircled V) refines particular construction steps.

cations of these models. For example, in [3, 8] models are used as virtual training systems in the medicine, while in [11] they are used for a construction of organ atlases. Although, above mentioned solutions are effective on fields for which they were developed, it can not be affirmed that they are general or generic. Also the construction procedure it is not generic. These models are not complete as well, because they cover just some viewpoints (e.g. just surgical planning and not actual virtual surgery).

3 Method: measurement, modelling, and simulation

In the sequel, we propose a procedure for construction of virtual models of the medical phenomena. Figure 1 depicts a construction procedure with all major building blocks. The construction of virtual medical model is guided from the initial step, i.e. acquisition of patient imaging material (Subsection 3.1), followed by the description of 3D reconstruction based on the image processing (Subsection 3.2), to the basic modelling-simulation approaches and directives (Subsection 3.3). It can be seen from Figure 1 that the results of verification/validation (denoted by the encircled letter V) influence the previous steps. They actually help to refine object detection, modelling, and simulation, and, consequently, constructed virtual medical model. This description is substantiated by example of the human knee joint virtual model construction. Results are taken from the SimBio project [15]. The SimBio project is an example of the proposed construction procedure defined in Figure 1.

3.1 Acquisition of imaging material

The first step in construction of virtual models is to acquire patient specific anatomy. This is usually achieved by non-invasive imaging methods. It is essential to acquire high-quality imaging material, because all other construction steps depend upon it (see Figure 1).

We deal with a human knee joint anatomy in our example. Figure 2 depicts the knee joint with three main bones– bone femur, bone tibia, and bone patella. Our intention is to observe these bones with their belonging cartilages and menisci (both lateral and medial), while all other structures in the knee joint are insignificant.

Important parts of patient anatomy indirectly narrow a set of potential medical imaging devices. The MR imaging technique was chosen to achieve our aim. We have access to the Toshiba Visart 1.5T MR scanner. Knee joint was imaged with different settings of the MR scanner parameters, such as image technique, flip angle, field of view (FOV), slice thickness, imaging timing parameters (TR and TE), number of acquisitions (NAQ), size of output matrix, etc. MR scanner parameters were selected in order to emphasise boundaries between bones, cartilages, and menisci, and, at the same time, retain adequate resolution for subsequent 3D reconstruction. It should be noted that altering scanner parameters always alters image quality, which is in proportion to the acquisition time. The selected MR scanner parameters were: FE3D image technique with QD knee coil, TR was 41 ms, TE was 9.0 ms, flip angle was fixed at 18/73, NAQ was 1, effective pixel size was 0.4 mm, FOV was 22 x 22 cm, and the output image matrix was 512x512 pixels. The number of images (slices or cross-sections) in the sequence was 60 with 2 mm slice thickness. Acquisition time was around 22 minutes.

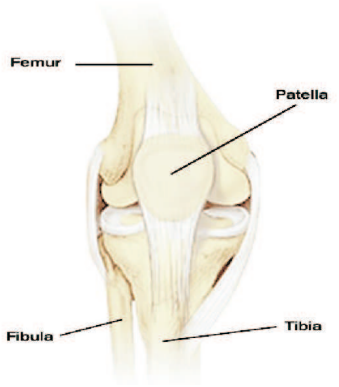


Figure 2: Human knee joint with ma3n bones.

3.1.1 Additional patient-specific information acquisition

If realistic object reconstruction is sought, then it is mandatory to keep sufficient spatial and lateral resolution of patient imaging material. Sometimes also other patient information is necessary. The reason can be twofold, namely, additional information can be mandatory for modelling procedure, or can be used for the verification/validation of

behaviour of constructed virtual models.

Our sample virtual medical model should imitate kinematics in the human knee joint. Additional information, as for instance the force in the knee, and examples of real knee kinematics are thus required. The patient should be examined when performing a gait cycle to acquire the most representative values. This can be achieved in an expensive open MRI scanner. Therefore, we seek an alternative solution. We designed a special MR compliant exercise rig to record the gait cycle and forces. The MR rig allows a volunteer or patient to undertake a controlled exercise protocol while exerting known light forces. This rig is fully MR compliant–it does not utilise any metal or ferromagnetic materials in its construction. Wooden part of this rig–i.e. pedal–was basically constructed according to [10] and, afterwards, modified to enable setting of 6 different knee flexion angle positions. Angles vary with respect to the patient's leg length, but generally are in the range from 0 to 40 degrees flexion with 8 to 10 degrees increments. Figure 3 (a) depicts this pedal. Particular knee angle position can be selected manually during MR imaging by adjusting a wooden coil. The pedal expanse is limited by the MR scanner bore dimensions. Maximal knee flexion angle supported is thus around 60 degrees.

Patient with flexed knee pushes against the wooden pedal during MR imaging. This force is measured with a special optical force measuring system which was also developed. A core of this system is a force sensor, which utilizes a simple principle of measuring the optical power losses in optical fibre caused by bending the fibre. The fibre multi-loop coil is positioned between two walls of the force sensor case (see Figure 3 (b)), separated by elastic spacers (rubber). The applied force causes displacement of the movable sensor case wall that bends the optical fibre coil and induces decreasing of the passed optical power. Alteration in optical power is afterwards transformed into a force value, which is displayed and stored by PC-based monitoring system. Figure 3 (c) depicts a volunteer during the acquisition of specific parameters by using the described MR-compliant exercise rig.

If additional patient-specific information except imaging material is required, then it is reasonable to simultaneously record all patient data and images. Possible errors in subsequent modelling are minimised in this way.

3.2 Image processing and 3D reconstruction

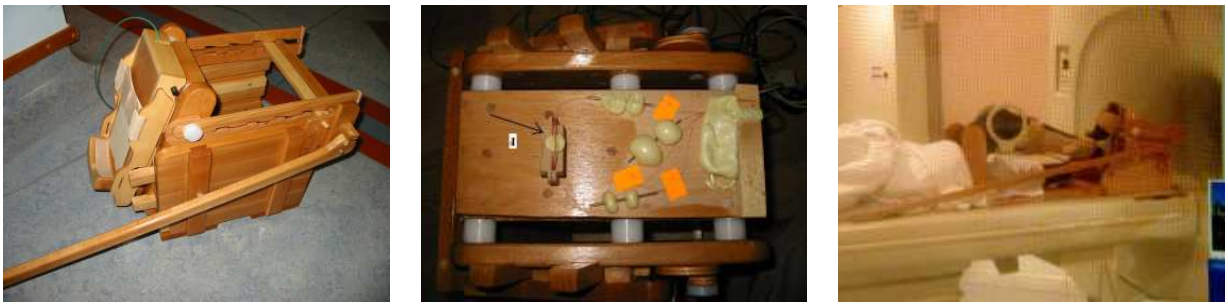
The next step after acquiring patient imaging material is to build 3D models of the observed organs or tissues (see Figure 1). This anatomy is usually obtained by applying 3D reconstruction on image segmentation results. The object recognition encompasses in general three major steps: preprocessing, segmentation, and object classification. Preprocessing either removes artefacts from images and/or enhances particular object features. The aim of image segmentation is to group pixels (voxels) with similar features into potential objects (bodies). Finally, an object classification discriminates between actual objects, background, and noise by using some criteria function.

There is variety of segmentation methods applicable for medical image sequences [14], either designed for 2D object detection in particular image (cross-section) from the sequence or 3D detection where image sequence is treated as a whole. Despite heterogeneity of methods, there are some directives applicable for method selection. First, a decision between automated or semi-automated segmentation should be taken. Although automated object recognition is preferred, it should be used with caution; namely, the obtained results must always be thoroughly verified. On the other hand, the semi-automated segmentation requires a clinician interaction during the processing, and verification is normally not necessary.

Segmentation methods are classified into three major groups [16]: a) pixel based or thresholding, b) contour or edge based, and c) region based methods. Selection of segmentation method depends primarily upon image modality (e.g. ultrasound, MR, positron emission tomography–PET) and the type of searched objects. If boundaries between objects and background are distinctive and well-defined, then methods from all three groups are potential candidates. If boundaries or edges are, however, weak, then thresholding and edge-based methods will not perform well. In real applications, the segmentation is a combination of all three major groups. On the other hand, the image modality defines the quality of boundary and potential artefacts (noise) in images. In ultrasound images, for instance, it is known that edges are not expressed and that speckle noise is present [13].

Object recognition procedures are designed to detect objects with special features. It is therefore common to include a prior-knowledge about objects and image modality in this recognition process. There are two possibilities for considering the prior-knowledge: a) segmentation method is context-based, b) classification is context-based. The first option means that segmentation method favours regions with pre-described shape and features (e.g. expected size, mean grey-level, compactness, and texture). On the other hand, the context-based classification extracts objects out of all regions obtained by segmentation according to some defined criteria. Criteria encompass prior-knowledge, while the segmentation method treats all regions equally in this process.

A big problem of object detection procedures is that they are not portable between medical imaging devices of same type but from different vendors (e.g. Philips MR scanner and Toshiba Visart MR scanner). Recognition process is usually developed by analysing the imaging material from a single device. The variability inherent when using different imaging devices is therefore not taken into account. Parameters of medical imaging device (see Subsection 3.1.1) are limited and, thus, theoretically could not be uniformed for all devices. Quality recognition process should therefore not be founded on grey-level features, but potentially on the contrast invariant features.



(a) (b) (c)

Figure 3: MR compliant exercise rig: a) wooden pedal, b) optical force sensor integrated on pedal, c) volunteer during additional parameter acquisition in the MR scanner.

The next step after segmentation is 3D object reconstruction. The reconstruction builds a 3D object model from partial segmentation results obtained on slices (cross-sections). Competent reconstruction supposes accurate segmentation and exact position of each slice in the 3D world or sufficient knowledge about image forming. Parallelism of subsequent cross-sections in the sequence greatly simplifies 3D reconstruction. In this case, it is mandatory to know exact inter-slice distance. If all conditions at image acquisition are not known or if even medical imaging device is not capable to return all these parameters, then the object reconstruction can only be informative. Figure 4 depicts a poor 3D reconstruction of two follicles from ultrasound ovarian image sequence obtained by intra-vaginal probe. This reconstruction mis-assumed that cross-sections are parallel; in fact, all cross-sections have common origin (i.e. probe) and an angular displacement between two subsequent cross-sections is constant. This displacement is, however, not possible to obtain from 2D ultrasound scanner used in this example. This indicates that not all medical problems could be accurately reconstructed.

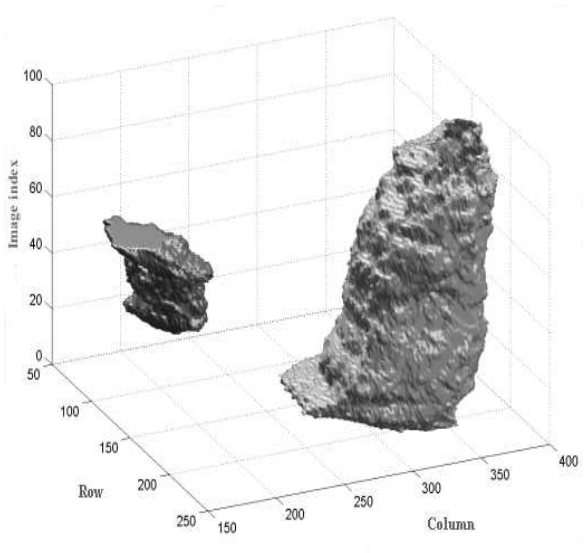


Figure 4: Poor 3D follicle reconstruction from 2D ultrasound ovarian image sequence.

3.2.1 Registration of MR knee images and 3D reconstruction

Image sequences from our human knee joint example are segmented by using a non-linear registration or mapping, respectively. Registration method was applied for two reasons: a) weak edges of knee structures, and b) simple and efficient integration of prior-knowledge (e.g. definition of relations between bones and cartilages, narrowing searching region). A patient knee image sequence is actually registered to a template knee image (sequence). Template knee image is constructed only once and is immutable in the registration process. It was constructed from knee image sequence of a typical patient. Each knee structure was accurately annotated on every MR slice. These readings actually define relations and approximate positions of bones, cartilages, and menisci in the human knee. The registration algorithm applied is based on the following registration equation [19]:

$$f - m = \frac{1}{2}[\triangle u(r)\frac{\partial f}{\partial u(r)} - \triangle u(r)^{-1}\frac{\partial m}{\partial u(r)}], \quad (1)$$

where f is the fixed or template image, m is the moved or patient image, $\triangle u(r)$ is the mapping function which maps m to f, and $\triangle u(r)^{-1}$ is the inverse function which maps f to m. Making the assumption that $\triangle u(r) \approx -\triangle u(r)^{-1}$ and gradient of $f \approx$ gradient of m, it is possible to reduce this equation to either of the following:

$$f - m = \triangle u(r)\frac{\partial f}{\partial u(r)} \quad \text{or} \quad f - m = -\triangle u(r)\frac{\partial m}{\partial u(r)}.$$

A quality measure for goodness-of-fit between both images is based on sum-of-squares of differences of voxel grey-level intensities. Full details of this registration routine will be published elsewhere.

This registration routine results in a mapping function (see Eq. (1)). Mapping function can be used to map readings (e.g. bone femur) from the template image sequence

to the patient image sequence. These partial results can be afterwards used for 3D reconstruction. Performing reconstruction this way is possible and valid, however, it introduce some artefacts. Disturbing staircase pattern (i.e. terracing problem) is frequently noticed in transitions from slice to slice. Another reconstruction approach was therefore followed. First, the 3D template knee model was developed from template image by using manual annotations. This template model was afterwards mapped by the calculated mapping function and, thus, the patient 3D knee model with sufficient quality is obtained.

Template knee model was constructed by using a combination of tools [15]. This knee model is actually finite element (FE) 3D mesh. First, the SURFdriver 3.5 software [18] was used to hand segment three bones, the belonging articular cartilage surfaces, and both menisci. The 3D surface points for each of these structures were then imported into Ansys meshing software [1], where meshes were manually refined. From a FE point of view the bones themselves are considered as rigid body structures. Bones were then passed from Ansys into Matlab environment, where the articular cartilage was generated using the registration algorithm described above. The outer surface of the bone structure was morphed onto the outer surface of the cartilage, and a set of 3D 8-node elements were created using this mapping. The smoothness constraints in the registration algorithm ensure that these elements have an acceptable geometry for analysis and simulation.

Described registration and mapping of template knee model produce high-quality patient-specific FE 3D knee meshes as shown in Figure 5.

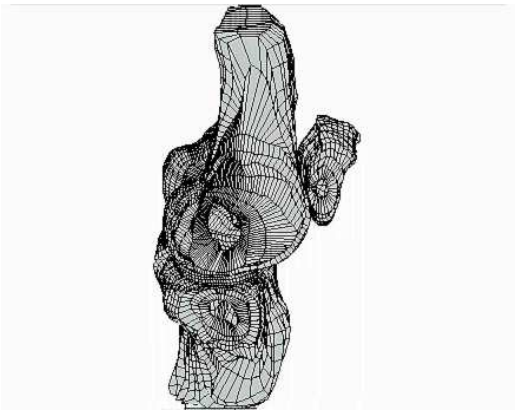


Figure 5: Patient-specific FE 3D knee mesh.

3.3 Modelling and simulation

Reconstructed human organs and tissues are usually presented in a form of surface meshes. Surface is formed from 3, 4 or 6 connected vertices, which consequently defines triangular, tetrahedral or hexahedral mesh. The aim of modelling phase (see Figure 1) is, however, to imitate the functionality of organs. An appropriate virtual model needs thus to be set. Many real-world properties must be linked to surface meshes of reconstructed objects. The information about the object's material properties (e.g. stiffness, friction) is mandatory. To imitate kinematics it is necessarily to prescribe the trajectory of movement for all objects and their interdependence. For complete virtual model it is also required to define scenarios of model behaviour in different situations.

Many general purpose modelling and simulation tools are available on the market to simplify this development (e.g. MSC.Software Suite, PAM Suite, ABAQUS). Besides, it is possible also to develop own modelling-simulation tools specially adjusted for specific problems–for instance a virtual delivery room simulator in [8]. Commercial tools are usually well tested and documented, with good technical support, and extensive consumer list. However, their drawbacks are usually limited options of a tool and, of course, high price. On the other hand, special designed software can support all options required, but on an expense of lengthy development and poorer validation/verification.

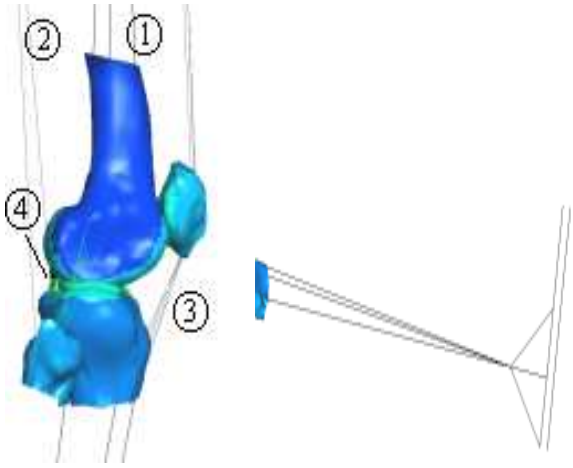
Virtual model of human knee joint was built by a commercial modelling/simulation software PAM Suite [4]. PAM Suite is actually a bundle of three products: a modelling tool Generis, a finite element problem solver or simulator Safe, and a visualisation tool View. The kinematics of reconstructed patient knee joint–represented in a form of FE surface meshes–was thus modelled by the PAM Generis. Four major sub-problems were addressed in the modelling phase: a) assigning appropriate material properties to each knee structure, b) design of accurate MR exercise rig model (foot-pedal), c) applying correct force on foot-pedal, and d) definition of the knee structure interdependence.

Several material properties like density, shear, yield stress, bulk modulus, Poisson's ratio, linear elastic stiffness, and coefficient of static friction are assigned to each knee structure. The parametric values were carefully selected [15]. It is known that some material properties are changed by temperature alteration and by patient aging. Nevertheless, we simplified our model and assigned the same property values to all patients.

A MR compliant exercise rig (see Subsection 3.1.1) was used during a patient imaging material acquisition. Its purpose was to mimic and partially record the conditions during a gait cycle (i.e. acquisition of forces and angles between the major knee bones). The simplified rig model as shown in Figure 6 (b) was added to human knee joint model. The foot-pedal is modelled by 3 and foot beams by 7 bars. A bar is a special element defined by two nodes and some material properties. All movements around these two nodes are possible. However, only forces in the direction of the bar can be applied.

Three main knee bones and fibula are defined as rigid bodies to avoid deformations. Their shape is completely described by the reconstructed FE surface mesh. Ligaments and muscles are manually added to the knee model.

They are used to define interdependence of bones and, consequently, enable the knee kinematics (i.e. knee extension/flexion). They are modelled as bars (see Figure 6 (a)). Quadriceps muscles and tendons (label 1) are modelled by 3 bars, while hamstring (label 2) is modelled by 2 bars, where one bar is connected to the fibula and the other on the tibia. Patellar ligament (label 3) consists of 5 bars linking the patella with the tibia. Lateral and medial collateral ligaments (label 4) are modelled by 4 bars on each side of the knee. They are main links between the femur and tibia (or fibula). Bars are also used to fasten menisci to the tibia. Without these bars, the meniscus remains in its initial positions during the simulation. Anterior and posterior cruciate ligaments are also modelled by bars. They ensure the knee stability.



(a) (b)

Figure 6: Finite element models: a) knee model, b) MR exercise rig model

The whole leg must be modelled to simulate a patient pushing against the foot-pedal of the rig. Missing parts of the bone femur and tibia are modelled by 3 bars each. These bars join in a common node, which enables rotation of both bones in every direction. MR exercise rig model (see Figure 6 (b)) is connected to the node, where bars representing the tibia meet. The footplate points are restricted to translate exactly as the foot is. To simulate the knee flexion during a gait cycle, it is necessary to apply a correct force on the foot-pedal. This force is concentrated in a point of the foot-pedal; its value was, however, obtained during the patient MRI examination.

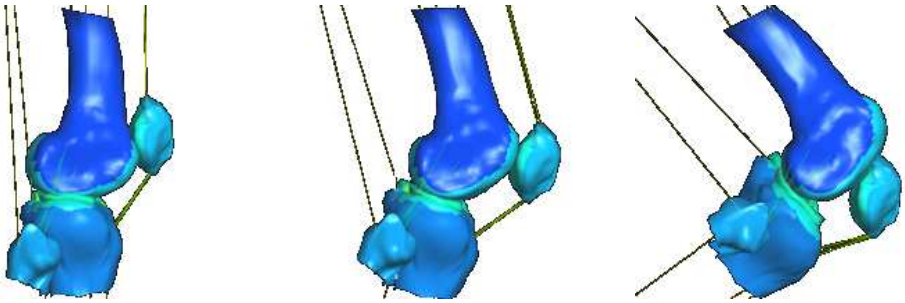
The modelling phase is followed by the simulation (see Figure 1). The obtained patient-specific knee model is afterwards simulated by using the PAM Safe tool. No manual interaction is required during the simulation. PAM Safe offers a very useful option to track the position of elements (e.g. node) or observe forces during the simulation. The simulation results in the simulated knee flexion ranging from fully extended knee to the knee flexed around 90 degrees. Results can be visualised by the PAM View tool.

4 Results

Efficiency of the built knee joint model will be presented in this section. This model actually simulates knee kinematics during a gait cycle under different scenarios. It is possible, for instance, to remove anterior crucial ligaments or simulate broken meniscus by the modelling tool PAM Generis and, consequently, observe knee kinematics of the altered knee anatomy. Results are visualised by the PAM View tool. It is possible just to observe knee kinematics in a form of animation movie with graphical manipulation option (e.g. rotation of the field of view, removing some uninteresting structures), or observe the results on a detailed level of nodes and forces. Figure 7 depicts the simulation results for the human knee joint model from its initial, extended, position to the final position of fully flexed knee.

There are 3 validation/verification milestones foreseen in the process of virtual model construction as seen from Figure 1. However, it is impossible to verify solely the modelling phase of our sample construction process. Verification is thus performed on 2 spots only–after image processing and simulation phase. Results after image processing (and reconstruction) phase are usually 3D surface meshes for particular structure. Figure 5 depicts surface meshes for reconstructed MR knee joint (see Subsection 3.2.1). These intermediate results must be compared to the correct real-world circumstances in order to assess the accuracy of reconstructed surface meshes and, indirectly, also the quality of segmentation-reconstruction process. The most competent verification relies on construction of virtual model for a phantom (e.g. phantom of the human knee joint). In the verification phase, the obtained model and also all intermediate results (e.g. surface meshes) are compared to the phantom. Usually, there are no phantoms available or they are very expensive. The most widespread verification technique is thus to compare the reconstructed surface meshes with the data provided by several experts. Experts usually manually annotate all important structures through entire image sequence. Afterwards, the so called "mean expert" annotations are calculated, thus minimizing inter-observer variability. These annotations are then used to build the experts' surface meshes, which are compared with the reconstructed surface meshes.

We performed twofold verification: a) 3D verification, where the reconstructed meshes are statistically compared to experts' (orthopeadic surgeons) meshes, and b) 2D verification, where the reconstructed meshes are first cut through and, the obtained contours are, afterwards, compared to experts' manual readings for a particular slice. The Haussdorf distance (HD), the mean absolute distance (MAD), and the spherical distance (SD) as a generalization of MAD distance for the 3D space, are measured between the experts' and segmentation-reconstruction results. Table 1 depicts the average distances calculated for surface meshes of 6 patients. Three major bones and their corresponding cartilages are verified. A bigger distance indicates a bigger error of the registration-reconstruction pro-



(a) (b) (c)

Figure 7: Simulated knee kinematics: a) initial knee position (simulation cycle 0), b) knee flexed around 45 degrees (cycle 150), c) fully flexed knee (cycle 300).

cess. The HD distance measures the biggest distance between two corresponding points from two curves, while the MAD distance returns an average distance between two curves. The HD distance is thus always bigger than MAD distance. The obtained results point out sufficient quality for subsequent construction phases. Also the visual inspection of these intermediate results performed by orthopeadic surgeons confirmed a good accuracy through entire image sequence (if the contours are observed). The biggest error is noticed on both extreme sides of a particular knee structure, where this structure begins to emerge or sink. These regions are very unexpressed and, therefore, present a huge problem for registration routine.

	MAD (mm)	HD (mm)	SD (mm)
BF	1.51	7.98	4.81
BT	1.48	7.34	3.66
BP	1.42	6.03	2.42
CF	1.32	9.41	7.62
CT	0.92	4.92	2.13
CP	1.22	6.07	2.65

Table 1: Statistical assessment of registration-reconstruction process for three bones (BF–femur, BT–tibia, and BP–patella) and their corresponding cartilages.

Calculated knee surface FE meshes are afterwards used in the modelling and simulation procedure as described in Subsection 3.3. Visual inspection of the final virtual human knee joint model performed by clinicians was focused on the relations and deformations of articulating bodies during the knee flexion. In general, the articular proportions in the simulated knee joint are very clear. No obvious and non-physiological structure deformations, with exception of slight patella gliding disturbances, are noticed.

As a quantitative verification measure of the model quality, three Euler rotation angles for bone tibia with respect to bone femur are observed–i.e. rotation around X axis (varus/valgus), around Y axis (internal/external rotation), and around Z axis (flexion/extension). It is known that during a gait cycle both major knee bones experience some translation and rotation with respect to their initial position. The measured rotations in the X, Y, Z directions of the tibia relative to the femur derived from the MR images are thus compared to the modelled rotations. Measured angle values are calculated from low-resolution MR knee images acquired by using MR exercise rig (see Subsection 3.1.1). Patient was additionally imaged in 6 flexion positions of knee. Image sequence acquired at a particular knee flexion angle was, afterwards, registered to bone femur and bone tibia surface meshes (volumes). This registration results in affine matrix which can be decomposed into required Euler rotation angles. Three Euler rotation angles can also be calculated from the simulation results. Twenty nodes from each major bone are traced during the simulation. For the i-th cycle, the following 4x20 matrix is defined:

$$\mathbf{X_i} = \begin{bmatrix} x_{i,1} & x_{i,2} & & x_{i,20} \\ y_{i,1} & y_{i,2} & & y_{i,20} \\ z_{i,1} & z_{i,2} & \cdots & z_{i,20} \\ 1 & 1 & & 1 \end{bmatrix},$$

where triplet $(x_{i,j}, y_{i,j}, z_{i,j})$ present coordinates for the j-th node in the i-th cycle. Usually, there was 300 simulation cycles. Then, the following predetermined linear equation system must be solved:

$$\mathbf{X_i} = \mathbf{A}_i \mathbf{X}_0,$$

where $\mathbf{X_0}$ denotes a matrix with the initial bone position (at cycle 0), $\mathbf{X_i}$ denotes a matrix with the bone position at cycle i, and $\mathbf{A_i}$ denotes the affine matrix in cycle i. After calculating affine matrices for all cycles for both bones, a new affine matrix of tibial flexion position with respect to femur is calculated in each cycle as follows:

$$\mathbf{J_i} = \mathbf{T_i^{-1}} \mathbf{F_i},$$

where $\mathbf{T_i}$ and $\mathbf{F_i}$ denote affine matrices in the i-th cycle for bone tibia and femur, respectively. This affine matrix takes the form of:

$$\begin{bmatrix} & [\mathbf{R}] & & [\mathbf{L}] \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

The Euler angles for each simulation have been calculated from the 3x3 rotation matrix, $\mathbf{R}$, and the translation vector, $\mathbf{L}$.

A comparison between the measured and Euler rotation angles obtained from the simulation pointed out a small difference in X and Y direction at fixed Z angle. This difference varied up to 10 degrees [15]. These results confirmed the virtual human knee joint model as very prospective.

Virtual knee model was built by using a PC-based system with the following configuration: 2 Intel Pentium Xenon 2.2 GHz processors, 1 GB RAM, and RAID 5 organisation of 120 GB hard discs. Processing accompanied by the reconstruction takes about 23 minutes on this PC system, modelling around 1 hour of manual work, while the simulation requires around 40 to 50 hours (300 simulation cycles). The later essentially depends upon specific patient anatomy, material properties, and force information used. The registration-reconstruction code is currently written just for single-processor computer systems; no speed up is thus expected if model construction is performed on multi-processor computer systems. Fortunately, there exist also multi-processor and cluster versions of the PAM Safe simulation tool. If simulation is run on 8-node cluster system, where each node has 2 processors, the calculated speed-up factor is then around 3 [15]. The most consumable construction part is thus reduced to 13 to 17 hours. The 8-node cluster system is optimal for our knee model consisting of a small number of FE elements. If a bigger cluster would be used, then execution times will be dominated by communication and no extra speed-up will be gained.

5 Discussion

Results of virtual human knee joint model were presented in the previous section. In the sequel, we will discuss the results, outline some problems and potential solutions for virtual model construction. Although discussion is bounded to human knee joint example, there are many suggestions and directives applicable to all similar applications.

Assessment of efficiency and accuracy of the constructed virtual medical model requires not only visual inspection, which provides just initial quality impression, but also thorough validation and verification. Figure 1 depicts that the construction of such models is a multi-step procedure. Thus, the final model depends upon all previous phases which must be evaluated as well. The error introduced in a particular processing phase is reflected also in the final model. It is very important to isolate potential sources of errors and to understand how this error is transferred through the processing chain. To achieve this, it is necessary to know the actual behaviour of the structure that is being modelled and the forces that it undergoes before it can be determined whether the behaviour of the model is valid.

The construction process begins with the patient-specific data. These data are usually considered completely valid. However, it is very common that some discrepancies occur during imaging. Errors introduced in this pre-phase are mainly neglected by the researchers, but such errors are extremely important because they can not be removed in subsequent phases. It is also very difficult to discover and correct these errors once imaging material is acquired and stored. A good idea is thus to record an entire process of patient material acquisition by a CCD camera, for example and, if in some later phase an error occurs in the imaging material, the entire acquisition process can be reviewed.

Construction of patient-specific knee joint model is not a completely automated procedure. Actually, it consists of two major building blocks (if we neglect imaging phase): registration-reconstruction part and modelling-simulation part. Image processing part is automated, while modelling phase requires user interaction. Modelling usually requires one hour of trained technician time. User must manually add some bars to the patient FE knee mesh and make some other corrections of the model. Fastening of elements (e.g. bars) to the mesh presents a potential source of errors, because the elements can be attached to wrong places. On the other hand, manual modelling is very flexible, as nearly any knee clinical case can be modelled (e.g. meniscus tear). However, from the verification point of view it is better that the construction tool is not so universal, but it supports a few checked construction scenarios. For a completely automated construction procedure, fastening points for ligaments and muscles should be detected in the image processing phase for each patient.

The registration routine used in the image processing phase (see Subsection 3.2.1) depends considerably upon MR scanner used in the imaging phase. It is known, that the variability inherent in MR imaging could cause the images of the same subject collected with the same MR protocol but with two different scanners may not be identical. Currently, this registration routine is fine tuned for the Toshiba Visart and Phillips Eclipse MR scanners. If knee joint images of other MR scanners are used, then an initialisation phase is required. This phase compensates variability found in the new type of images with regard to template image. This phase should theoretically be done just once per new type of MR scanner.

Modelling is the least deterministic phase in the entire process, because only a small portion of this phase is automated (e.g. assigning the material properties to elements), while the majority requires manual interaction. Modelling is to some extent intuitive (especially ligament and muscle fastening) and highly dependent upon the experts' knowledge and experience. This phase can not be verified directly. The only feedback is verification of simulation results. However, the verification just points out that there are some problems in the model, but it does not isolate sources of errors. In the current model, the verification is based just upon Euler rotation angles for bone tibia with respect to bone femur. It is thus suggested to expand verification also on the behaviour of other knee structures.

Stability of the patient FE knee mesh used in the modelling process is also very dependent upon the template mesh morphing and, indirectly, on the quality of the en-

tire image registration. For example, if the medial meniscus has been deformed considerably for particular patient, possibly due to actual pathology within the meniscus, the morphing algorithm will have to reduce the element sizes markedly. This by all means will not aid simulation stability. Therefore, it is possible that another manual intervention into modelling process may be necessary for gross knee pathologies.

In the sequel, the virtual human knee joint model is discussed from a medical point of view. Verification of the image processing part pointed out adequate accuracy of detected anatomical knee structures, especially in the orthopaedic surgeons target regions where the most pathologies occur (i.e. the condylar or articular and intercondylar region). The first benefit of knee model is thus an easy visualisation of detected knee structures in a particular cross section, even in anatomically problematic regions. In some cases, this "second opinion" could be of great help for clinicians in a preoperative planning and decision.

The simulated knee joint kinematics is almost realistic. In general, articular proportions in the simulated knee joint are very clear and no physiological structure deformations were detected. There are only some slight disturbances of the patella gliding at the extreme knee flexion, which might cause some disturbances in the patella kinematics. The shape and position of menisci are also very clear and no meniscus surface deformations are seen. The clinical benefit of the virtual knee joint model is also a better visualisation of the patient knee kinematics. For example, the knee could seem from MR images at first sight clinically stable, while the simulation points out enough functional instability. This was the case for a patient with partial anterior crucial ligament rupture, where suspicion on the ligament rupture was indicated only by the virtual knee model and confirmed by an arthroscopy examination. The knee instability was so big that the operative treatment was necessary.

Virtual knee joint model could have a big significance for planning operative interventions. It could be especially advantageous in the situations where postoperative knee joint stability and functionality is not obvious. When replacing a meniscus, for instance, a size of meniscus implant is selected by rule of thumb. Appropriateness of choice is usually confirmed about one year after operation. Thus, it is much better to play through different scenarios by using virtual model when taking such decisions.

The described virtual knee joint model has not been used in daily clinical practice yet. For such usage, this virtual model must be accompanied by several auxiliary tools. Visualisation tools and tools for correcting intermediate results are indispensable. Such tools should, for instance, visualize image registration results (i.e. 3D patient-specific knee mesh) and enable manual correction of particular segmentation results and, consequently, 3D knee mesh. Simulation results are currently visualized in the PAM View tool. The trained personnel only can interpret these results. More sophisticated model should also be accompanied by a tool for interpreting results from the medical point of view.

6 Conclusion

The framework for construction of patient-specific virtual medical models was presented in this paper. All construction blocks from imaging, image processing, modelling, and simulation were described and applicable directives issued. This framework was successfully applied by construction of virtual model of the human knee joint. Statistical assessment of the developed knee model pointed out sufficient accuracy of intermediate results and the final knee kinematics as well. This model was assessed also by the clinicians as very prospective.

Acknowledgement

This work was supported by the European funding in the 5th Framework project entitled SimBio (Contract No. IST-1999-10378).

References

[1] Ansys Inc., http://www.ansys.com/

[2] I.N. Bankman (2000), *Handbook of medical imaging: Processing and analysis*, Academic Press.

[3] C. Basdogan, C.-H. Ho, M.A. Srinivasan (2001), Virtual environments for medical training: graphical and haptic simulation of laparoscopic common bile duct exploration, *IEEE transactions on mechatronics*, vol. 6, no. 3, pp. 269–285.

[4] ESI group, PAM Suite, http://www.esi-group.com/

[5] E.M. Haacke, R.W. Brown, M.R. Thompson, R. Venkatesan (1999), *Magnetic resonance imaging: Physical principles and sequence design*, Wiley-Liss.

[6] J.V. Hajnal, D.L.G. Hill, D.J. Hawkes (2001), *Medical image registration*, CRC Press.

[7] T. He, L. Hong, D. Chen, Z. Liang (2001), Reliable path for virtual endoscopy: ensuring complete examination of human organs, *IEEE transactions on visualization and computer graphics*, vol. 7, no. 4, pp. 333–342.

[8] D. Korošec, A. Holobar, M. Divjak, D. Zazula (2003), Multilevel implementation of the dynamic virtual environment, *Proceedings of Fifth International Conference on Simulations in Biomedicine*, WIT Press, Southampton, pp. 477–486.

[9] J.-D. Lee, C.-H. Huang, S.-T. Lee (2002), Improving stereotactic surgery using 3-D reconstruction, *IEEE engineering in medicine and biology*, vol. 21, no. 6, pp. 109–116.

[10] A.D. McCarthy, D.R. Hose, D.C. Barber, S. Wood, G. Darwent, D. Chan, D.R. Bickerstaff, I.D. Wilkinson (2003), A registration-based MR method for calculating in-vivo 3-D knee joint motion: Validating finite element simulations, *Proceedings of the International society for magnetic resonance in medicine*, ISMRM, Toronto.

[11] H. Park, P. H. Bland, C.R. Meyer (2003), Construction of an abdominal atlas and its application in segmentation, *IEEE transaction on medical imaging*, vol. 22, no. 4, pp. 483–492

[12] B. Potočnik, D. Zazula (2001), Assessing the efficiency of the image segmentation algorithms, Electrotechnical review, vol. 68, no. 2/3, pp. 97-104.

[13] B. Potočnik, D. Zazula (2002), Automated analysis of a sequence of ovarian ultrasound images, Part I: Segmentation of single 2D images, *Image vision and computing* vol. 20, no. 3, pp. 217–225.

[14] B. Potočnik, D. Zazula (2002), Automated analysis of a sequence of ovarian ultrasound images, Part II: Prediction-based object recognition from a sequence of images, *Image vision and computing*, vol. 20, no. 3, pp. 227–235.

[15] SimBio project, http://www.simbio.de

[16] M. Sonka, V. Hlavac, R. Boyle (1994), *Image processing, analysis and machine vision*, Chapman and Hall.

[17] W. Sun, P. Lal (2002), Recent development on computer aided tissue engineeringŰŰa review, *Computer methods and programs in biomedicine*, vol. 67, no. 2, pp. 85–103.

[18] Surfdriver software, http://www.surfdriver.com/

[19] S. Wood, D.C. Barber, A.D. McCarthy, D. Chan, I.D. Wilkinson, G. Darwent, D.R. Hose (2002), A novel image registration application for the in vivo quantification of joint kinematics, *Proceedings of medical image understanding and analysis*, University of Portsmouth, Portsmouth.

[20] J. Xia, H.H.S. Ip, N. Samman, H.T.F. Wong, J. Gateno, W. Dongfeng, R.W.K. Yeung, C.S.B. Kot, H. Tideman (2001), Three-dimensional virtual-reality surgical planning and soft-tissue prediction for orthognathic surgery, *IEEE transactions on information technology in biomedicine*, vol. 5, no. 2, pp. 97–107.

System Resource Utilization Analysis based on Model Checking Method

Ki-Seok Bang, Hyun-Wook Jin, Chuck-Yoo and Jin-Young Choi
Department of Computer Science & Engineering, Korea Univerity
{kbang, choi}@formal.korea.ac.kr
{hwjin, chuck}@os.korea.ac.kr

Keywords: Model Checking, Temporal Logic, Property Specification, SPIN, LTL, Myrinet NIC

Received: July 21, 2004

Model checking method is a widely used formal method for proving whether or not a given model satisfies properties, and for producing counter examples if the model does not satisfy properties. In this paper, we show model checking methods can be used for resource utilization analysis of systems. We specify system utilization properties using temporal logic called LTL, and find a bottleneck of system performance using model checking.

Povzetek: Analiza uporabe sistemskih virov z metodo preverjanja modelov.

1 Introduction

Formal methods[5] are the most notable efforts to guarantee a correctness of system design and behaviors. Correctness of design is a very important factor of H/W and S/W systems for preventing an economical and human losses caused by minor errors. Especially, model checking[5] is one of the most active research areas because its procedures are automatic and easy to understand. In model checking, we model a system as a finite state machine and specify the properties that must be satisfied by the real system using a temporal logic[12]. After that, we automatically perform a model checker whether the system satisfies its properties or not. In general, properties are mostly describing correctness or safety of the system's operation. It is very important to specify the correctness property of system design and behavior, and an appropriate property must be specified to represent a correct requirement.

However, in some cases, the correctness property is not an important factor for system designers. In a small system such as NIC(Network Interface Card), the correctness can be ignored by the designer and user. Instead of the safety characteristic, a system efficiency such as operating speed or performance of system resource utilization is more important to evaluate the system's quality level.

Generally, measurement or simulation is used to show an efficiency property of a system. A professional analyst measures responses that occur during an experiment using a simulator, and compares them to an ideal computed value[15]. If the two values are similar or equal, then it can be said the system uses its resources effectively and shows a high performance. Otherwise, the conclusion is that the system performs ineffectively. Then a reason of the ineffectiveness should be found.

To detect it, they must analyze both H/W and S/W. Especially, they must inspect the whole source codes for S/W analysis. However, it is almost impossible to analyze source codes perfectly, since the codes are too long and cooperates with other systems in a complicate manner. In addition, network systems are constructed in a distributed environment, so error detection is very difficult even if source codes are instpected.

In this paper, we specify system utilization properties using temporal logic, and show that model checking methods can be used for performance analysis. We used the model checker SPIN[8, 9] and LTL(Linear Temporal Logic)[12] to perform this research. We analyzed a Myrinet NIC firmware system[3] and successfully found a reason for ineffective behavior of the system.

This paper is composed as follows; Chapter 2 is a brief introduction to network simulation tools and firmware design methodologies, and model checking method. We address the extension of LTL specification to a quantity characteristic in Chapter 3. In Chapter 4, we explain an overhead analysis of high speed network card and results of model checking. We conclude in Chapter 5.

2 Related Works

2.1 Simulation of network status and firmware desin methods

Currently, certain network simulators like Network Simulator II(NS-II)[15] are used to design network system or analyse their behavior. This simulator performs a simulation for TCP, routing, and multicast protocols on wire/wireless network. We can find and fix bugs in the network protocol and communication software using the simulator.

But, simulators just provide a convenient method for users or software designers to fix their software codes, since the system modelled by a simulator is not a real system, but an ideal model. And, NS-II has many of its own bugs and errors. Besides that, users must check whether

the errors from a simulation are from simulator itself.

The research to design NIC firmware correctly is progressing. An improved NIC program for high performance MPI of INRIA modelled behaviors of NIC firmware using a state transition diagram[17]. They modelled and analyzed behaviors of the NIC sender and receiver are working in parallel using the state transition diagram. In this way, it can be helpful for analyzing complex send/receive behaviors of firmware.

2.2 Model Checking

Model checking[5] is an automatic verification technique for correctness of finite state systems. That is a process to prove a correctness of system through logical proving about system constraints or requirement for safe system behavior. Even model checking has many advantages and disadvantages, but can verify a complex system as a hardware circuit or communication protocol automatically. Because verification processes are performed automatically, so the verification results are correct and easy to analysis. In addition, model checking is performed to whole states of system state space, it can conclude yes or no for very large system.

The process of model checking is as follows : The first task is to convert a system to a formal model accepted by a model checker. In practice, this process is not automated and formal languages defined by formal semantics must be used to specify a system. Many abstraction techniques are applied to draw a abstract model in this process. Abstraction is very important for reducing states of a system because system space can be exploded during the model checking process. After modeling, we needs to specify properties that the system must hold. The specification usually is given in some logical formalism. Generally, temporal logics are used to represent a temporal characteristic of systems. The verification is completely automatic with the abstract model and properties. However, it does need human assistance to analyze the result of model checking. The model checker can produce a counterexample for the checked property, and it can help the designer in tracking down where the error occurred. In this case, analyzing the error trace may require a modification to the system and reapplication of the model checking process. The error can also result from incorrect modeling of the system or from an incorrect specification. The error trace can also be useful in identifying and fixing these two problems.

There are many representative model checkers; SMV[11] and SPIN[8, 9] are two examples. SMV is a CTL model checker for hardware verification, and SPIN is an LTL model checker for communication protocol or concurrent software. In this paper, we specify system properties using LTL and verify them using SPIN.

SPIN is a representative LTL model checker. SPIN supports its own tools for LTL specification and verification. A User can model a system by Promela, an input language of SPIN, and specify a required property in LTL. Then SPIN verifies the model and generates verification results, “True” or counterexample if the result is “false”. SPIN provides some basic safety and liveness properties such as deadlock, invalid end state and non-progress cycle. Therefore, we don’t have to specify those properties. SPIN also has many optional switches, so we can control search spaces and verification times. We can simulate the model’s behavior using SPIN simulator before verification. The simulation facility can reduce the verification time and human efforts to specify a complex property.

2.3 Temporal Logics

It is very important to specify system property that certain system must be satisfied. In general, CTL(Computational Tree Logic) or LTL(Linear Temporal Logic) is used for property specification of model checking. Two logics specify behaviors of a system according to time structure. Time is assumed to have a branching structure in CTL. That is, it models system behavior using state graph that represents a infinite state transition tree from the initial state. LTL assumes that the time sequence is linear, the system’s behavior is represented by one linear sequence[12].

Model checking can be divided into CTL model checking and LTL model checking based on the used temporal logic. In CTL model checking, we model finite state system using Kripke structure and prove the temporal logic is satisfied on an arbitrary state of this system by fixed point theory. LTL model checking models systems and LTL properties using automata, and checks the emptiness of two automata[5].

3 Extension of property specification using temporal logic

In general, model checking is used to prove the correctness or safety of systems, and property specification by temporal logics represents that kind of requirement. Especially, deadlock and invalid endstate are the most common safety properties. so model checkers can check those properties without temporal logic specifications. For example, it is very important to verify mutualy exclusive behaviors in the critical section of operating system design. When two concurrent processes are trying to operate in a critical section, we can specify their mutual exclusive property as follows; $[]\,!\,(P \wedge Q)$. This logic can be translated into " Always P and Q cannot be true at the same time," and means two processes cannot operate in a critical section at the same time.

Another property, “if one process tries to be in a critical section, then eventually that process can operate in the critical section.” can be specified for liveness requirement. This property can be written: $[](P \rightarrow <> Q)$. Safety and liveness properties are very important to guarantee a system’s behavior and can find many implicit errors easily. However, in some cases, the correctness property is not an im-

portant factor for system designers. In a small system such as NIC(Network Interface Card), the correctness can be ignored by designer and user. Instead of the safety characteristic, a system efficiency such as operating speed or performance of system resource utilization is more important to evaluate the system's quality level. Generally, measurement or simulation is used to show an efficiency property of system. A professional analyst measures responses that occurr during an experiment using a simulator, and compare them to an ideal computed value. If two values are similar or equal, then it can be said the system uses its resources effectively and shows a high performance. Otherwise, the conclusion is that the system performs ineffectively. Then a reason for the ineffectiveness should be found.

To detect it, they must analyze both H/W and S/W. Especially, they must inspect the whole source codes for S/W analysis. However, it is almost impossible to analyze source codes perfectly, since the codes are too long and cooperates with other systems in a complicate manner . In addition, network systems are constructed in a distributed environment, so error detection is very difficult even if source codes are inspected. However, it is too difficult to find a reason of error and fix it in the program source code. In this paper, we show an easy way to analyze source code using model checking.

In fact, a direct verification of quantitive property is not so easy. Therefore, simulation or performance measurement must be used with verification to increase the possibility of finding an error.

First, we simulate a target system to measure its performance, and analyze its measurements to compare system's effectiveness. If it is too low, we can assume the system has some problems. Then, we model the system using some abstract techniques, and specify a property which must be satisfied to the system using a temporal logic. In this case, the property specification must be concerned with a performance not a safety. For example, we can specify a resource sharing characteristic by checking if two processes can be moved to a specific state simultaneously. That is, two processes can operate with the same shared resource at the same time. If the model checking result shows false, it means resource sharing is impossible and system performance can be dropped. Of course, we should guess the kind of errors. But this method is faster and more correct than the traditional code inspection. We can find and fix a problem by a logical proof.

4 Example of system resource utilization analysis using model checking

4.1 Performance analysis of high-speed network card

We performed model checking for performance analysis of Myrinet NIC(Network Interface Card)[3].

Gigabit network interface cards(NIC) like Myrinet are becoming popular. In order to achieve the best possible performance out of Myrinet, several user-level communication primitives have been proposed[4, 7, 16]. Berkeley-VIA[4] is a well-known implementation of Virtual Interface Architecture (VIA)[6] that is an industrial standard for user-level communication primitives. Therefore, it is generally expected that VIA can achieve near physical bandwidth of gigabit networks. However, our research shows that

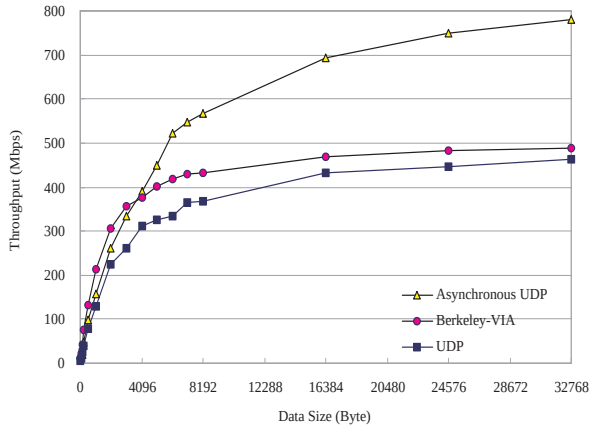


Figure 1: Throughput comparison of Asynchronous UDP, Berkeley-VIA, and UDP

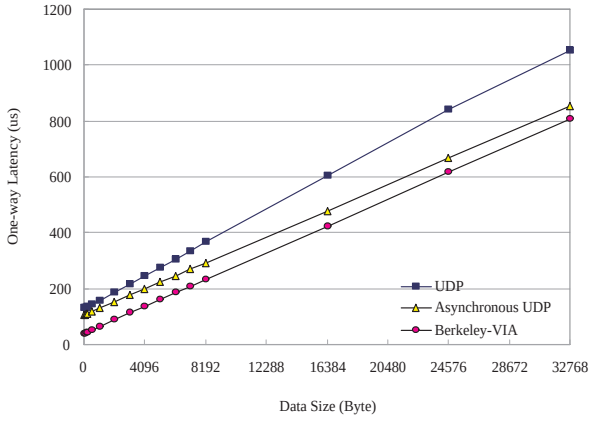


Figure 2: One-way latency comparison of Asynchronous UDP, Berkeley-VIA, and UDP

Berkeley-VIA is able to achieve a slightly higher throughput than UDP on Myrinet as shown in Figure 1. Furthermore, Berkeley-VIA has much less throughput than an improved UDP named Asynchronous UDP[18]. On the other hand, we find that Berkeley-VIA has the shortest one-way latency as shown in Figure 2, which indicates that Berkeley-VIA has less communication overhead than UDP and Asynchronous UDP.

So the question is why Berkeley-VIA has a very low overhead but is not able to achieve the best possible

throughput. Our goal is to find the performance bottleneck. The firmware of Myrinet NIC needs to be analyzed to see where the bottleneck is. Because Myrinet NIC has three DMA engines and separate memory and CPU, the firmware itself is very complicated. Therefore, the analysis of the firmware is not an easy task. Also the interaction between the firmware and the host is very complex so that the firmware analysis becomes even more complicated.

Therefore, we first build state transition diagrams of the firmware in order to analyze the firmware of Myrinet NIC. Second, we translate the state transition diagrams into specifications written in PROMELA (PROcess MEta LAnguage)[8]. Third, we derive verification formulas, and then the formulas are verified with SPIN.

4.2 Myrinet Network Interface Card

Myrinet is a gigabit Local Area Network(LAN), which supports full-duplex 1.28+1.28 Gbps bandwidth[1, 2]. In this section, we describe the hardware components of Myrinet NIC based on LANai-4[13]. Myrinet NIC consists of a RISC processor named LANai, Static Random Access Memory (SRAM), and three DMA engines. As shown in Figure 3, LANai executes the firmware, and SRAM stores the data for sending or receiving. Each DMA engine works as follows.

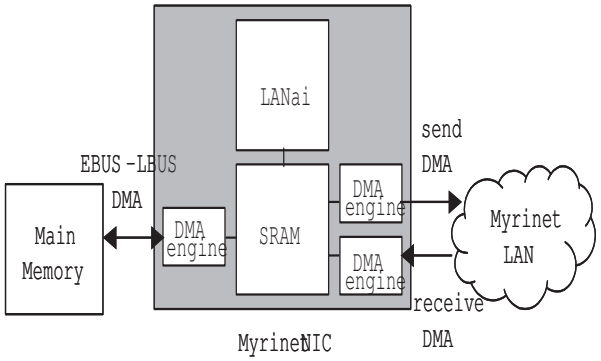


Figure 3: Hardware feature of Myrinet NIC

The EBUS-LBUS DMA engine is responsible for the data movement between the main memory and the SRAM. The send-DMA engine moves the data in SRAM to the Myrinet physical network.The receive-DMA engine receives data from Myrinet LAN into the SRAM. The firmware initiates the DMA operations by setting the proper registers of each DMA engine and notices the completion of corresponding DMA operation via the 32-bit Interrupt Status Register (ISR) on LANai processor[3].

4.3 Modeling of firmware

In this section, we discuss the modeling of LCP and MCP. We construct the state transition diagrams for concerned modules based on their source codes and specify them with PROMELA.

4.3.1 LCP

LCP is the firmware for Berkeley-VIA. LCP consists of four modules: hostDma, lcpTx, lcpRx, and main. Figures 4and 5 show the state transition diagrams of former three modules. The hostDma module is responsible for

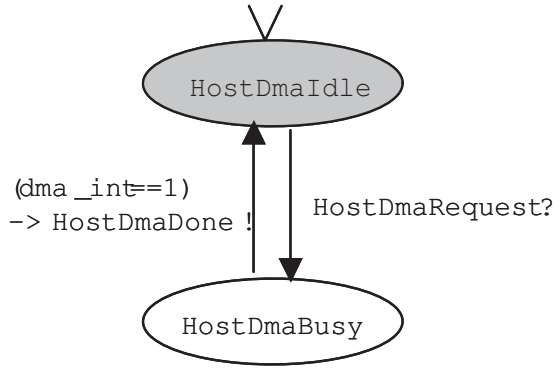


Figure 4: State transition diagram of the hostDma module

EBUS-LBUS DMA. The initial state of the hostDma module is HostDmaIdle. The lcpTx and lcpRx modules invoke the method of the hostDma module. Then, the hostDma module initializes the EBUS-LBUS DMA operation, and its state moves to HostDmaBusy. When the EBUS-LBUS DMA operation is done, the state of the hostDma module moves from HostDmaBusy to HostDmaIdle, and the method returns to its invoker.

The lcpTx module sends data and lcpRx receives data. Their initial state is LcpTxIdle and LcpRxReady. Each module invokes a method of HostDma at the initial state. If data is received from the network during the send-DMA operation, the lcpTx module invokes the method of the lcpRx module and moves to the LcpTxInvokeRx state. When the lcpRx module has received a data completely, its method returns to the lcpTx module, and the state of the lcpTx module moves to LcpTxSendDma again.

Note that the entry point of the hostDma, lcpTx, and lcpRx modules is the initial state of each module. We will discuss the entry point in the next section, compared it with

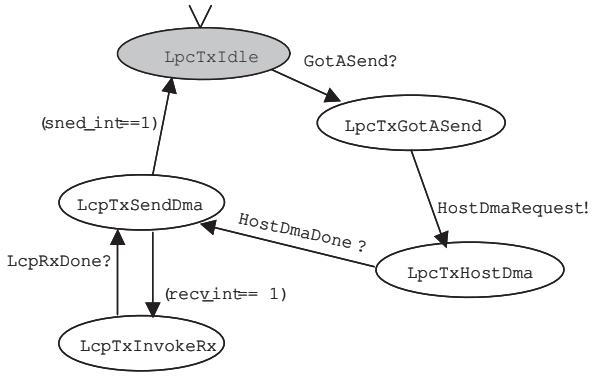


Figure 5: State transition diagram of the lcpTx module

MCP.

We specify the modules as processes in PROMELA. All invocations between modules are performed in a synchronous manner.

4.3.2 MCP

MCP is included in the Myrinet Software package[1] While Berkeley-VIA supports only VIA protocol, Myrinet Software supports TCP/IP protocol suite. MCP consists of five modules: hostSend, netSend, hostReceive, netReceive, and main. The hostSend module moves data from main memory to SRAM, and the netSend module sends the data in SRAM to the network. The netReceive module receives the data from the network to SRAM. The hostReceive module moves the received data to the main memory. The state transition diagrams of four modules are shown in Figures 6,7,8 and 9.

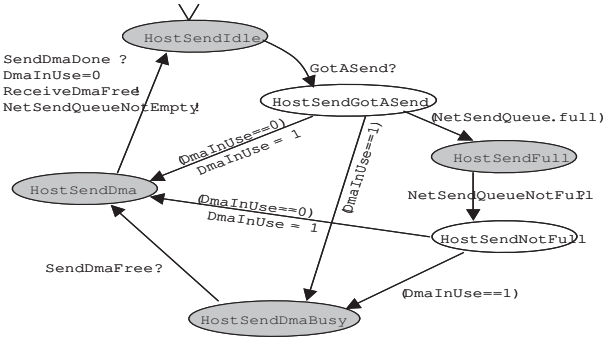


Figure 6: State transition diagram of the hostSend module

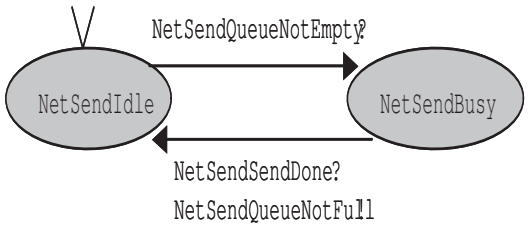


Figure 7: State transition diagram of the netSend module

Compared with modules of LCP, the notable difference is that each module of MCP has plural entry points. This means that the method of each module is invoked from an entry point and returns when it moves to another entry point without waiting for the next event. Therefore, the method invoked in the next time starts from the state in which the method returns right before. On the other hand, in the case of LCP, a method is invoked when the module is in the initial state and returns only when it goes back to the initial state. we implement an invocation by using two rendezvous communication channels of PROMELA. For more details, refer [10].

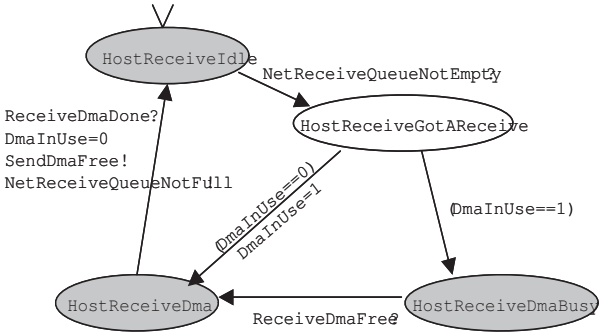


Figure 8: State transition diagram of the hostReceive module

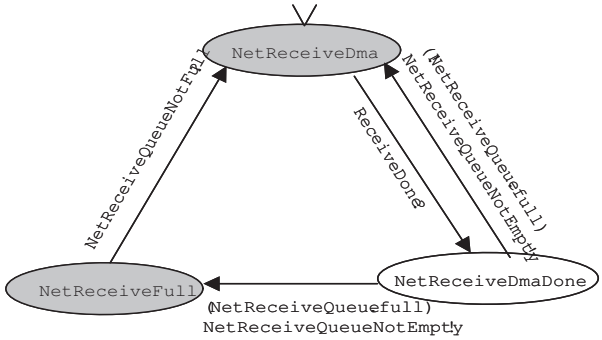


Figure 9: State transition diagram of the netReceive module

4.4 Formal verification of firmware behaviors

First, we verified the correctness of each firmware using SPIN. The correctness means only one module should occupy the EBUS-LBUS DMA engine at a time. The EBUS-LBUS DMA engine moves data not only from main memory to SRAM for sending, but also from SRAM to main memory for receiving. Therefore, if the other module already occupies the DMA engine, a module should wait until the DMA engine becomes idle. The formulas used are as follows:

1. LCP

   [] ! (LTHD && LRHD)

   The lcpTx module cannot use the EBUS-LBUS DMA engine while the lcpRx module occupies it.

2. MCP

   [] ! (HSD && HRD)

   The hostSend module cannot use the EBUS-LBUS DMA engine during the hostReceive module occupies it.

The hostSend module cannot use the EBUS-LBUS DMA engine while the hostReceive module occupies it.

The verification results show that both LCP and MCP satisfy the above correctness property.

```
Full statespace search for:
 never-claim         +
 assertion violations + (if within scope of claim)
 acceptance cycles  - (not selected)
 invalid endstates  - (disabled by never-claim)

State-vector 492 byte, depth reached 472,
errors: 0
```

Table 1: Formal verification result of NIC - Safety property

However, we can find a significant difference between two firmwares when analyze the behaviors of LCP and MCP from the viewpoint of throughput. The key factor that determines the throughput of NIC is how well the DMA engines are utilized. The maximum throughput can be achieved when the EBUS-LBUS DMA engine performs in parallel with the send-DMA and receive-DMA engine.

For example, let $\mathrm{DMA}_{\mathrm{EBUS-LBUS}}$ be the throughput of the EBUS-LBUS DMA and $\mathrm{DMA}_{\mathrm{send}}$ be the throughput of the send-DMA. $\mathrm{DMA}_{\mathrm{EBUS-LBUS}}$ is determined by the bandwidth of the I/O bus (e.g. PCI) that connects the main memory and SRAM of NIC. On the other hand, $\mathrm{DMA}_{\mathrm{send}}$ is determined by the network physical media. When DMA engines perform in parallel, the throughput is evaluated as follows:

$$Throughput = MIN(DMA_{EBUS-LBUS}, DMA_{send})$$

However, if DMA engines perform sequentially, the throughput is limited as follows:

$$Throughput = DMA_{EBUS-LBUS} / (1 + DMA_{EBUS-LBUS}/DMA_{send})$$

If $\mathrm{DMA}_{\mathrm{EBUS-LBUS}}$ and $\mathrm{DMA}_{\mathrm{send}}$ are the same, the throughput achieved is reduced to 1/2 of $\mathrm{DMA}_{\mathrm{EBUS-LBUS}}$. The next step of the analysis is to derive verification formulas. Because the verification formulas need to reflect the utilization of DMA engines, we use the following formulas written in LTL:

1. LCP
   A. $\diamond$ (LTIR && HDB && ! LRHD)

      Can the lcpTx module initiate send-DMA while the hostDma module is using EBUS-LBUS DMA that moves data from main memory to SRAM?

   B. $\diamond$ (LRR && HDB && ! LTHD)

      Can the lcpRx module initiate receive-DMA while the hostDma module is using EBUS-LBUS DMA that moves data from SRAM to main memory?

      - LTIR : The state of lcpTx is LcpTxInvokeRx.
      - LTHD : The state of lcpTx is LcpTxHostDma.
      - LRR : The state of lcpRx is LcpRxReady.
      - LRHD : The state of lcpRx is LcpRxHostDma.
      - HDB : The state of hostDma is HostDmaBusy.

2. MCP
   A. $\diamond$ (HSD && NSB)

      Can the netSend module initiate send-DMA while the hostSend module occupies the EBUS-LBUS DMA engine?

   B. $\diamond$ (HRD && NRD)

      Can the netReceive module initiate receive-DMA while the hostReceive module occupies EBUS-LBUS DMA engine?

      - HSD : The state of hostSend is HostSendDma.
      - NSB : The state of netSend is NetSendBusy.
      - HRD : The state of hostReceive is HostReceiveDma.
      - NRD : The state of netReceive is NetReceiveDma.

If DMA engines perform in parallel, each verification formula should result in "True." When we run SPIN with the above formulas, the verification formulas of MCP are "True." However, the formulas for LCP result in "False." That is, LCP cannot perform the send-DMA during the EBUS-LBUS DMA that moves data from main memory to SRAM (formula 1-A).

```
Full statespace search for:
 never-claim         +
 assertion violations + (if within scope of claim)
 acceptance cycles  - (not selected)
 invalid endstates  - (disabled by never-claim)

State-vector 492 byte, depth reached 472,
errors: 1
```

Table 2: Formal verification result of LCP - Reource Utilization property

In order to quantify the parallelism of each firmware, we measure the DMA overheads. Figures 10 and 11 are the time charts. The x-axis indicates the time, and the y-axis represents the DMA engine that each packet goes through in order to be processed. A rectangle in a DMA engine is the time spent in the DMA engine to process a packet. A rectangle starts at the time when the corresponding rectangle in upper DMA finishes. Figure 9 shows that the DMA overheads of MCP are fully overlapped, while the overheads of LCP cannot be pipelined at all, as shown in Figure 11. Also LCP cannot perform the receive-DMA as

well during the EBUS-LBUS DMA that moves data from SRAM to main memory (formula 1-B). This result explains why the performance of Berkeley-VIA is limited. The simulation results also show that LCP performs DMA sequentially but MCP performs DMA in parallel. We have run random and interactive simulations and confirmed the same results.

We could prove the safetiness of NIC firmware and find a reason which causes a problem in the performance of NIC. If we perform a simulation to find performance problems, it is easy to show a falling-off in performance. However, we must analyze the source code of firmware to find the reason. The real source code of NIC is very complex, as well as too long to analyze. Therefore, we can perform performance analysis and find a reason of performance dropping using model checking.

5 Conclusions

This paper explains a way to identify the problems which caused system performance drop using model checking.

Generally, logical formula used in model checking is related to system correctness or safety. It is important that systems have to be designed accurately for system safety, since system's error affects not only the system but its environment. But as shown in this paper, even already verified systems showed the lower processing rate. In that case, we can conclude that system resources are used ineffectively.

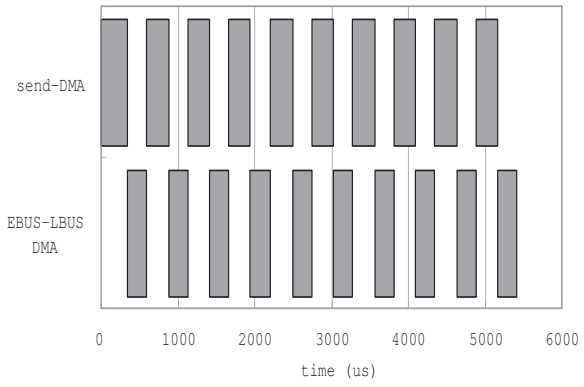


Figure 10: DMA overheads of LCP

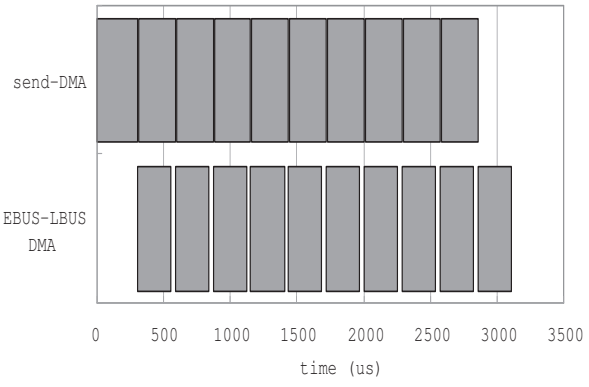


Figure 11: DMA overheads of MCP

Usually, the quantitative analysis such as performance of system and usability of resources is done by simulation or measurement. As a result, it is very difficult to analyze the cause of performance drop.

In other words, the source code must be analyzed directly to find the major factor. However, as systems get complicated and interactions with other systems increase, it is impossible to directly analyze the source code. In this case, we propose to adapt a model checking for quantitative analysis like resource utilization. If quantitative property can be specified by temporal logic, then quantitative analysis as performance analysis could be performed easily. Also, if a reason for performance drop could be found easily in an abstract model, applying it with the actual code to fix the source code would be an easy task.

Still, there are a few obstacles to adapting the model checking to performance analysis. First, the model checking can express finite systems like hardware controller or communication protocol as a finite state machine, but other systems can have infinite states, so they have to be modelled by special abstract methods. During the abstraction process, there could be many new and potential errors. The most obvious problem to perform the model checking is the state space explosion problem. Though this method can search every possible state in finite state system, the states can be exploded when the system is performing. Therefore, it is impossible to search all possible state space. Thus depending on the capability of computer which performs model checking and verification algorithm, the number of states that can be verified is limited. To solve this weakness of model checking, there are many attempts and much research to formally verify the finite states of the complicated system and automation of abstraction process to analyzation of source code. If more effective model checking methods could be developed by this research, verification of safety and analyzation of performance of large and complicated systems can be performed easily.

In this study, we performed a performance analysis using SPIN. In the future, there are plans to test the effectiveness with more model checkers. Also, based on the example of NIC, there should be similar research done with bigger and more performance sensiteve systems, for example, embedded systems.

References

[1] D. Anderson, J. Chase, S. Gadde, A. Gallatin, K. Yocum, and M. Feeley, Cheating the I/O Bottleneck: Network Storage with Trapeze/Myrinet, Proceedings of the 1998 USENIX Technical Conference, June 1998.

[2] T. E. Anderson, D. E. Culler, D. A. Patterson, and the NOW Team, A Case for Networks of Workstations: NOW, IEEE Micro, February 1995.

[3] N. J. Boden, D. Cohen, R. E. Felderman, A. E. Kulawik, C. L. Seitz, J. N. Seizovic, and W. -K. Su, Myrinet – A Gigabit-per-Second Local-Area Network, IEEE-Micro, Vol. 15, No. 1, pp. 29-36, February 1995.

[4] P. Buonadonna, A. Geweke, and D. Culler, An Implementation and Analysis of the Virtual Interface Architecture, Proceedings of SC'98, November 1998.

[5] E. M. Clarke, O. Grumberg, D. A. Peled, Model Checking, MIT Press, 1999.

[6] D. Dunning, G. Regnier, G. McAlpine, D. Cameron, B. Shubert, A. M. Berry, E. Gronke, and C. Dodd, The Virtual Interface Architecture, IEEE Micro, Vol. 8, pp. 66-76, March-April 1998.

[7] T. V. Eicken, A. Basu, V. Buch, and W. Vogels, U-Net: A User-Level Network Interface for Parallel and Distributed Computing, Proceedings of 15th ACM SOSP, pp. 40-53, December 1995.

[8] G. J. Holzmann, Design and Validation of Computer Protocols, Prentice Hall, 1991.

[9] G. J. Holzmann, The Model Checker SPIN, IEEE Transactions on Software Engineering, May 1997.

[10] H. W. Jin, K. S. Bang, J. Y. Choi, C. Yoo, Bottleneck Analysis of a Gigabit Network Interface Card, Proceedings of 9th International SPIN Workshop, pp. 170-185, May 2002.

[11] K. L. Macmillan, Symbolic Model Checking, Kluwer Academic Publishers, 1993.

[12] Z. Manna, A. Pnueli, The Temporal Logic of Reactive and Concurrent Systems, Springer-Verlag, 1992.

[13] Myricom Inc., LANai 4, http://www.myri.com, February 1999.

[14] Myricom Inc., Myrinet User's Guide, http://www.myri.com, 1996.

[15] The Network Simulator, http://www.isi.edu/nsnam/ns.

[16] L. Prylli and B. Tourancheau, BIP: a new protocol designed for high performance networking on myrinet, Proceedings of IPPS/SPDP98, 1998.

[17] L. Prylli and B. Tourancheau, R. Westrelin, An Improved NIC Program for High-Performance MPI, Proceedings of Workshop on Cluster-Based Computing, 1999.

[18] C. Yoo, H. -W. Jin, and S. -C. Kwon, Asynchronous UDP, IEICE Transactions on Communications, Vol.E84-B, No.12, December 2001.

Towards Neural Network Model for Insulin/Glucose in Diabetics-II

Raed Abu Zitar
College of Information Technology
Philadelphia University
Jordan
rzitar@philadelphia.edu.jo

Abdulkareem Al-Jabali
College of Engineering
Al-Isra University
Jordan

Keywords: Levenberg-Marquardt Neural Network, Polynomial Networks, Diabetics, Insulin

Received: March 18, 2004

In this work we extending our investigations for a general neural network model that resembles the interactions between glucose concentration levels and amount of insulin injected in the bodies of diabetics. We use real data for 70 different patients of diabetics and build on it our model. Two types of neural networks (NN's) are experimented in building that model; the first type is called the Levenberg-Marquardt (LM) training algorithm of multilayer feed forward neural network (NN), the other one is based on Polynomial Network (PN's). We do comparisons between the two models based on their performance. The design stages mainly consist of training, testing, and validation. A linear regression between the output of the multi-layer feed forward neural network trained by LM algorithm (abbreviated by LM NN) and the actual outputs shows that the LM NN is a better model. The PN's have proved to be good static "mappers", but their performance is degraded when used in modelling a dynamical system. The LM NN based model still proved that it can potentially be used to build a theoretical general regulator controller for insulin injections and, hence, can reflect an idea about the types and amounts of insulin required for patients.

Povzetek: Na osnovi podatkov o 70 pacientih je razvit nevronski model za razmerna med insulinom in glukozo.

1 Introduction

Diabetes is a disease in which the body cannot properly use the energy it gets from food. Normally, most of the food we eat is broken down or digested into sugar or glucose. Glucose provides the body's cells with the energy they need. Insulin, a hormone produced in the pancreas, helps the glucose get inside the cells where the glucose is burned for energy. In diabetes the body cannot make enough insulin or is resistant to the insulin it makes. As a result, your blood glucose can become much higher than usual. A normal fasting blood glucose range is about 65 -110. When your blood sugar is 126 or higher after fasting for eight hours, the diagnosis of diabetes is made.

It is a widespread chronic illness that accounts for a large part of the health care budget. It affects approximately one hundred million people world wide [1] and may lead to a variety of vascular, neurological or metabolic complications.

Diabetes and complications associated with it can be viewed as a partial or total failure of one or more intrinsic therapeutic feedback loops. In a healthy person the relationship established between glucose level and insulin secretion is an effective feedback control loop. Increased blood glucose level (the controlled variable) results in the production of the hormone insulin by the pancreas (the controller). This insulin reduces blood glucose from its elevated level. Diabetic patient has not this inter-relationship or it does not work as it does in healthy people.

In practice, the full picture is more complex and the diabetic patient needs to be regarded as a multi-input/multi-output physiological system which contains several controllable and measurable variables as well as other factors which are not directly observable. The patient's diet (the carbohydrate content of which will directly elevate blood glucose level), hormones (gastrointestinal, glucagons, …etc), the physical effort exerted, the amount of insulin delivered, and other factors [2] can be considered to be control variables which need to be adjusted in order to maintain homeostasis within the human organism. Obviously, the manipulation of all variables that affect the dynamics of diabetes is cumbersome.

1.1 Mathematical Models of Glucose/Insulin Dynamics

Mathematical models have provided one mean of understanding diabetes dynamics. There are various models based on glucose and insulin distributions, and those models have been used to explain glucose /insulin interaction . All these models are valid under certain conditions and assumptions [3]-[9]. These models represent a range of approaches, including linear [2],[3], nonlinear [4],[5], probabilistic [6], compartmental [7], non-compartmental [8], and parametric models [9]. Although these models may be useful in a research setting, they all have limitations in predicting blood glucose in real-time clinical situations because of the inherent requirement of frequently updated information about the models' variables like glucose loads and insulin availability. For example, glucose challenges to the body, such as those resulting from a meal, are important glucose sources in models, but are not conveniently measurable and must instead be considered as unknown disturbances. As another example, the timing and amount of subcutaneous insulin injections are known to the patient, but the resulting vascular availability of insulin is often variable, depending on factors such as the insulin dose and delivery site. Since frequent insulin determinations are not practical for routine management, only estimates of vascular insulin concentrations can be incorporated in models when applied in an actual clinical setting. In the absence of accurate, frequently updated information about glucose loads and insulin concentration, these conventional models can only be marginally effective in real time at reliably predicting future blood glucose values [10]. Given this situation, if continuous or very frequent blood glucose monitoring is available, recent and past glucose values may be exploited as an alternative to the use of conventional models to describe blood glucose dynamics.

The features of data that can be used for such studies are sometimes based on individual blood glucose values from a patient or a group of patients, while in many other studies statistical averages of repeated challenges for a given patient a or group of patients are used. Furthermore, blood glucose is sampled frequently enough to capture a detailed record of excursions. The monitoring period for a given individual is extended over a long time period (several weeks). Full information about external factors such as meals, insulin injections and the type, exercise, etc.. that cause blood glucose perturbations is also recorded.

2 The Neural Based Models

Feed forward neural networks have been used extensively to solve many kinds of problems, being applied in a wide range of areas covering subjects such as prediction of temporal series, structure prediction of proteins, and speech recognition [7]. One of the fundamental properties making these networks useful is its capacity to learn from examples. Through synaptic modifications algorithms, the network is capable of obtaining a new structure of internal connections that is appropriate for solving a determined task.

The general underlying theory of the whole learning process is poorly understood. There are few general results, especially concerning generalization. One particular point of interest is the selection of a concise subset of examples from the whole training set as a way of improving generalization ability. This problem has also been referred to as "active learning" or "query-based learning" by many authors. In a broad sense, these terms refer to any form of learning in which the learning algorithm has some control over the inputs used for the training.

In this work, we use two different types of neural networks; the LM NN model and the polynomial network model (PN's). In a previous work, we studied the modeling through Radial Basis Function Networks [?]. We showed that LM NN had more success than Radial Basis Networks. We related that to capability of LM NN training algorithm which is an advanced version of back propagation algorithm to capture the dynamics of the control surface the associates the patient state variables with the output which is the amount of insulin injected. Although, both of them are feed forward types of neural networks, they fundamentally differ in the way training is implemented. LM NN model is a feed forward model consisting of two layers. Its learning strategy starts with incremental error back propagation algorithm and gradually switches to conjugate gradient-based back propagation for the final convergence phase [11].

In the other hand, PN's technique is known for fast convergence toward "closest" local minimum and can escape shallow local minima. We may consider the problem of finding the proper amount of insulin as an identification problem which involves finding the best matching class given a list of target classes (and their models obtained in the training phase) In general, the training data for each class consists of a set of previous state variables of the possible input vectors that come from the history of the patient(s). In our case, each observation is represented by a single vector containing four previous values of the patient state variables which are present glucose level, previous glucose level, meals, exercises, short term insulin, medium term insulin, and long term insulin, we will come to the details of the simulations later .

Now, for each class, i, we have a set of N_i training observations represented by the N_i feature vectors $[\mathbf{x}_{i,1} \quad \mathbf{x}_{i,2} \quad \cdots \quad \mathbf{x}_{i,N_i}]^t$ Identification requires the decision between multiple hypotheses, Hi. Given an observation feature vector x, the Bayes decision rule [7] for this problem is

$$i^{opt} = \arg\max_i p(H_i \mid \mathbf{x}) \tag{1}$$

A common method for solving equation (1) is to approximate an ideal output on a set of training data with a network. That is, if $\{f_i(x)\}$ are discriminant functions [8], then we train $f_i(x)$ to an ideal output of 1 on all in-class observation feature vectors and 0 on all out-of-class

observation feature vectors. If f_i is optimized for mean-squared
error over all possible functions such that

$$f_i^{opt} = \arg\min_{f_i} E_{\mathbf{x},H}\left\{\left[f_i(x) - y_i(\mathbf{x},H)\right]^2\right\}, \quad (2)$$

The solution entails that:

$$f_i^{opt} = p(H_i \mid \mathbf{x})$$

In equation (2), $E_{x,\,H}$ is the expectation operator over the joint distribution of x and all hypotheses, and $yi(\mathbf{x},H)$ is the ideal output for H_i. Thus, the least squares optimization problem gives the functions necessary for the hypothesis test in equation(1). If the discriminant function in (9) is allowed to vary only over a given class (in our case polynomials with a limited degree), then the optimization problem of equation (9) gives an approximation of the a posteriori probabilities[8]. Using the resulting polynomial approximation in equation (8) thus gives an approximation to the ideal Bayes rule. The basic embodiment of a K_{th} order polynomial network consists of several parts. In the training phase, the elements of each training feature vector, x = [x_1, x_2 ..., x_M], are combined with multipliers to form a set of basis functions, *p(x).* The elements of *p(x)* are the monomials of the form:

$$\prod_{j=1}^{M} x_j^{k_j}, \text{ where } k_j \text{ is a positive integer, and } 0 \le \sum_{j=1}^{M} k_j \le K \quad (3)$$

Once the training feature vectors are expanded into their polynomial basis terms, the polynomial network is trained to approximate an ideal output using mean-squared error as the objective criterion. The polynomial expansion of the i*th* class feature vectors are denoted by:

$$\mathbf{M}_i = [\mathbf{p}(\mathbf{x}_{i,1}) \quad \mathbf{p}(\mathbf{x}_{i,2}) \quad \cdots \quad \mathbf{p}(\mathbf{x}_{i,N_i})]^t$$

The global matrix for all C classes is obtained by concatenating all the individual Mi matrices such that:

$$\mathbf{M} = [\mathbf{M}_1 \quad \mathbf{M}_2 \quad \cdots \quad \mathbf{M}_C]^t$$

The training problem reduces to finding an optimum set of weights, w, that minimizes the distance (in this case the in the L2 sense) between the ideal outputs and a linear combination of the polynomial expansion of the training data such that:

$$\mathbf{w}_i^{opt} = \arg\min_{\mathbf{w}} \left\|\mathbf{M}\mathbf{w} - \mathbf{o}_i\right\|_2$$

where $\boldsymbol{o_i}$ represents the ideal output comprised of the column vector whose entries are N_i ones in the rows where the *ith* class's data is located in M and zeros otherwise. The weights (identification models) $\boldsymbol{w_i^{op}}$ can be obtained explicitly (non iteratively) by applying the normal equations method [9] such as

$$\mathbf{M}^t\mathbf{M}\mathbf{w}_i^{opt} = \mathbf{M}^t\mathbf{o}_i$$

If we define:

$$\mathbf{R}_j = \mathbf{M}_j^t\mathbf{M}_j, \quad \mathbf{R} = \sum_{j=1}^{C}\mathbf{R}_j, \text{ and } \mathbf{m} = \mathbf{M}_j^t\mathbf{1}$$

this will yield:

$$\mathbf{w}_i^{opt} = \mathbf{R}^{-1}\mathbf{m} \quad (4)$$

In the recognition stage when an unknown feature vector, x, is presented to all C polynomial networks, the vector is expanded into its polynomial terms ***p(x)*** (similar to what was done in the training phase) and its class, c, is determined such that

$$c = \arg\max_i \mathbf{w}_i^{opt}\mathbf{p}(\mathbf{x}) \quad (5)$$

2.1 Simulations with Neural Networks

In our simulations, we used a set of data for 70 different patients. Sample of the data used is shown in Table (1). The terms; STI stands for short term insulin , MTI for midterm insulin, LTI for long term insulin. In the columns for exercise and meal, "1" stands for "yes" and "0" stands for "no". The terms PGL stands for present glucose level and NGL stands for next glucose level. The period of time is the minutes between two consecutive measurements of the glucose level in blood. However, we normalized data before training ending up with 0 mean and unity standard deviation. We did spectral component analysis and eliminated all components less than 0.1% of the variations. The components of a training vector in our data were the PGL, STI , MTI, time period, and meal. We eliminated the all "1" exercise input, the all "0" postprandial input, and the all "0" LTI input. These inputs have no effect since they do not contribute to the variation of the output as they are always kept constant to a single value. The single output of our model has a target of the NGL. This NGL is measured after the given time period of time. We had data for more than 70 patients with total of more than 30,000 samples of input/target training pairs. The training process itself is equivalent to a nonlinear regression process between the normalized inputs (spectral components) and the normalized targets. When training is complete, the output of the neural network is un-normalized in a reverse process for the principal components normalization stage that was implemented before training. The un-normalized data is then passed

Table. 1: Sample of patients data used for modelling

PGL mg/dL	STI U	MTI U	LTI U	Exercise	Meal	Postprandial	Time period (minutes)	NGL mg/dL
100	9	13	0	1	0	0	478	119
119	7	0	0	1	1	0	343	123
123	0	0	0	1	1	0	524	216
216	12	13	0	1	1	0	561	211
211	7	0	0	1	1	0	869	257
257	11	13	0	1	0	0	600	129
129	7	0	0	1	1	0	867	239
239	14	14	0	1	1	0	558	129
129	0	0	0	1	1	0	299	340

through a linear regression stage. The linear regression is implemented between the un-normalized outputs of the neural network and the actual targets taken from the data files (NGL). The linear regression reflects the degree of accuracy and correctness of the neural network predictions.

The training data were accessed as follows; for every consecutive four training points, the first and third point are used for training, the second point is used for testing, and the fourth point is used for validation. Then, the process is repeated for the whole set of data. Of course, during testing and validation there is no learning (training), only nonlinear regression through the neural network followed by a linear regression stage between targets and un-normalized outputs to measure accuracy of prediction.

It should be mentioned here that what is being done in this work is some kind of system identification [13], [14]. Our ultimate goal is to find some general parameters that govern the behavior of the glucose levels in diabetics. When some quantity of medication is investigated its crucial to search for a general theoretical model that can be used to help in testing the effect of that medication. Models such as the ones we present here can be used in giving a theoretical hint about the effect of the insulin in diabetics. These models can be further used in building insulin controllers that automatically insert the proper amount of insulin and work as regulator control for a required level of glucose in blood.

2.2 Simulations with Polynomial Network Model

The PN model we explained earlier is used to model the data of the 70 patients. This model architecture has one neuron at the output layer, see Figure.1. The number of neurons (units) at hidden layer starts with one, then two , and goes up as long as the error values did not reach the given criteria. The PN model (which could be considered as special type of neural networks) number of neurons at the output layer equal to the discrete ranges of insulin injections. The inputs are polynomialized , as we will see later, and then are treated as feature vectors that require classification to the right level of output class. This process is equivalent to a nonlinear layer in standard neural networks. The output layer only contains the weights that are associated with each class. Each neuron at the output layer is associated with a class as target. The error is calculated as the sum of squares between the output and the target divided over the sum of target values (in order to give percentage as shown in Table. 2.

The model we have here could not learn to predict correctly the next values of glucose levels (NGL). As a result of the previous experiments, PN's are only good "mappers" , as evident from Table. 2. results.

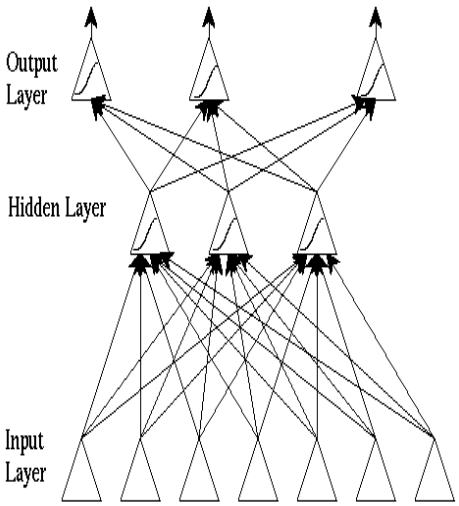


Figure. 1. A scheme of feed forward Neural Network (NN) that could be used with Levenberg-Marquardt (LM) training algorithm or even PN's under some assumptions.

2.3 Simulations with the Levenberg-Marquardt (LM) NN Model

In this model, we used 5 hidden units and one output unit. Adaptive parameters are used in calculating adjustments in weights and biases [15], [16]. Error back propagation algorithm in conjunction with Levenberg-Marquardt (LM) optimization [11] is used. This usually results in fast but memory consuming training. Figure. 2. shows graphs for training, testing, and validation. The training data is prepared in a manner similar to the previous method. The testing and the validation points in the graph are done by passing the inputs through the neural network only without any modifications for weights. The mean square error , which is the performance criteria, is calculated according to the difference between the target and the output of the neural network. It is clear from Figure. 2. as training error goes down, the testing and validation error also goes down.

Table2: Sample of PN's error rates.

group of observations	training error	testing error
1	14.6%	20.6%
2	16.7%	23.1%
3	21.1%	26.7%
4	22.3%	24.7%
5	12.5%	16.6%
6	10.5%	15.7%
7	11.56%	14.5%
8	16.8%	19.8%
9	21.6%	30.3%

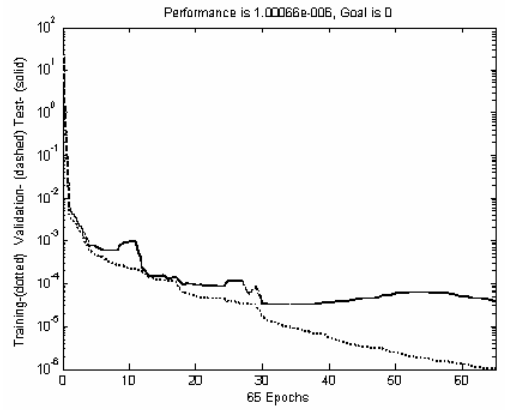


Figure. 2: The error versus Training/Validation/Testing epochs for Levenberg-Marquardt neural network.

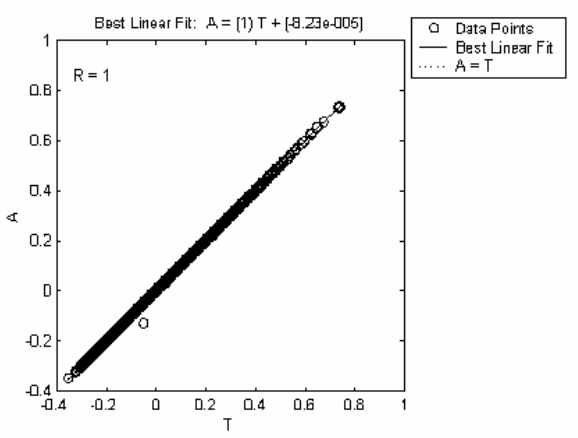


Figure 3. Linear regression of LM NN simulations

Figure. 3. shows a linear regression for the whole set of data. Although, around half of the data is only used in the training, the linear regression for the whole set of data is excellent. Also, note that, the linear regression is an outside process used only to map the normalized output of the neural network with the actual target data. However, the whole process of testing and validation is based on non linear regression. Neural networks are highly nonlinear by nature. The results demonstrate the ability of this type of networks to model the whole set of data. The neural network, here, could capture, identify, and generalize the insulin/glucose dynamics for the samples of the 70 patients with high accuracy. The normalization process for the raw inputs/targets has great effect on preparing the data to be suitable for the training. Without this normalization training the neural networks would have been very slow.

3 Conclusions and Discussions

RBF networks and Back propagation Feed forward networks have been applied with success to function approximation problems [17]. However, PN's were mainly used for pattern classification and recognition problems [20],[21], In this work, we propose using this type of networks in , a more like, regulator control problem. The idea as a whole is a decision making problem, in which a group of previous observations for the patients state variable are used to approximate a decision value for the amount of Insulin required. As shown in Table. 2., It is clear that testing error rates and even training error rates are higher than the acceptable. Again, LM NN is proved to be superior over PN's for this type of problems. The PN's even gave worse results when compared with Radial Basis Function Networks of our previous work [20]. The solutions derived by PN's come from numerical solutions of the polynomialized inputs matrices. Therefore, there is not much flexibility and adaptation in this method to catch up the severe nonlinearity and time dependency of this problem. The weakness could be in the mapping process in which the inputs were polynomialized. This process is similar to the kernel functions used by RBF's, however it is still less flexible in shape and works much more in general scale than the very local kernel functions used by the RBF network. The LM NN, on the other hand, adjusts all the parameters of the network at every training sample and hence, all parameters of the network contribute to the generation of the output concurrently. This would give LM NN more ability to create a global fit for data. Moreover, this collective behavior reduces the size of the network to much smaller size than that for PN. As a result, it is more advantageous to use LM NN when data is "expensive" (i.e. not abundant) and when data is complex. While it is advised to use PN's (or RBF networks) when the data is cheap or plentiful like in adaptive control or some signal processing applications [19]. PN's have the advantage of being fast in training especially when number of classes is small. As explained earlier, PN's has a "single shut" solution. It may suffer from ill-conditioned cases if the matrices were not singular, but this is rare to happen since inputs vary significantly. LM NN training process is more complicated and time consuming.

If we try to relate the results we have with the nature of data we are dealing with, it is fair to conclude that the nature of data we have is not an PN type of data. The target for training, which is the NLG, is not only a function of current state of patient and of the amount and type of insulin she/he just has, but also it is dependent on previous states of the patient and on previous medications she/he already has. The LM NN model is a successful method to identify and capture those dynamics. Some other techniques for modeling are based some on conceptual mathematical modeling followed by standard numerical optimization to approximate the model parameters (least squares method for example). However, in this paper we are more interested in Artificial Intelligence-based models and, in particular, in Neural Networks (NN's). Moreover, we presented two techniques of NN, one is more successful in classifying features. While (LM NN) has a more global strategy at which all hidden neurons participate in generating the output for some input or stimuli. As a matter of fact, NN proved to be a potentially good modeling tool for such type of problems, and that is the bottom line for this work. But we do not advice to use PN's in modeling of dynamical problems due to the limited success we had. We have decided even not to apply linear regression for

the outputs and the targets in the case of PN's due to the apparent unsuccessfulness.

Future work will include designing neural based controllers to regulate the level of glucose in blood based on those NN plant models. We hope that these neural network based techniques will add a little knowledge toward the understanding of insulin/glucose dynamics.

4 References

[1] E.Carson et al, , 1992 "A Spectrum of Approaches for Controlling Diabetes", IEEE Transactions on Control Systems, No.12.

[2] Pank, Klaus;Jurgens, Clemens; et al; , 1998 "Predictive neural networks for learning the time course of blood glucose levels from the complex Interaction of counter regulatory hormones" Neural Computation, Vol. 10 Issue 4.

[3] Bergman RN, Ider YZ, Bowden CR, Cobelli C, 1979: Quantitative estimation of insulin sensitivity. Am. J. Physiology 236:E667-E677.

[4] Lehmann ED, Hermanyi I, Deutsch T. , 1994 Retrospective validation of a physiological model of glucose-insulin interaction in type 1 diabetes mellitus. Med Eng Phys16:193-202, 1994 , Trans. Blamed Eng 41:116-124.

[5] Naylor JS, Hodel AS, Albisser AM, Evers JH, Strickland JH, Schumacher DA, 1997: Comparison of parameterized models for computer-based estimation of diabetic patient glucose response. Med. Inform. 22:21-34.

[6] Bremer, Troy; Gough, David A. ;1999 "Is blood glucose predictable from previous values? " Diabetes, Vol. 48 Issue 3.

[7] Hassoun M. H. ,1995 " Fundamentals of Artificial Neural Networks", MIT Press, Cambridge, Mass.

[8] Sturis J, Polonsky KS, Mosekilde E, Van Cauter E, 1991: Computer model for mechanisms underlying ultra oscillations of insulin and glucose. Am. J. Physiology 260:E801-E809.

[9] Andreassen S, Benn JJ, Hovorka R, Olesen KG, Carson ER, 1994: A probabilistic approach to glucose prediction and insulin dose adjustment: description of a metabolic model and pilot evaluation study. Computer Methods Programs Biomedical 41:153-163.

[10] Ferrannini E, Smith JD, Cobelli C; Toffolo G, Pilo A, DeFronzo RA, 1985: Effect of insulin on the distribution and disposition of glucose in man. J. Clinical Invest 76:357-364.

[11] Cobelli C, Toffolo G, Ferrannini E, 1996: A model of glucose kinetics and their control by insulin, compartmental and non compartmental approaches. Math Biosciences 71:291-316.

[12] Rumelhart, D. E., McClelland J. L., and the PDP Research Group, 1986. "Parallel Distributed Processing: Exploration in the Microstructure of Cognition." Vol.1. MIT Press, Cambridge, Mass.

[13] T. Kohonen, , 1993, "Self-organizing maps: Optimization approaches, in Artificial Neural Networks" ,T. Kohonen, K.Makisara, O.Simula, and J.Kanga, eds., pp..1147-1156. IEEE, New York.

[14] B. Eisenstein and R.Vaccaro, , 1982"Feature Extraction by System Identification," IEEE Trans. on Systems, Man, and Cybernetics, vol.SMC-12, No. 1, pp.42-50.

[15] Jefferies, C. , 1991" Code Recognition and set selection with neural networks" Boston, Birkhauser.

[16] Kosko, B. ed. , 1991 "Neural Networks for Signal Processing." , Prentice-Hall.

[17] Broomhead, D.S., and Lowe, D. Multivariate Function Interpolation and Adaptive Networks, Complex Systems, 2, 321-355.

[18] Kohonen, T. , 1983 "Self-Organization and Associative Memory," Springer-Verlag Series in Information Sciences 8.

[19] Lee, Y. , 1991 "Handwritten Digit recognition Using k- nearest Neighbor, Radial-basis Functions, and Back-propagation Neural Networks, Neural Computation, 3(3), 440-449.

[20] Abu Zitar, RA, 2004, " Towards Neural Networks Model for Insulin/Glucose in Diabetics," accepted at *the International Journal of Computers and Information Systems*.

[21] K. Fukunaga, 1990, Introduction to Statistical Pattern Recognition. Academic Press.

A Survey of Contemporary Real-time Operating Systems

S. Baskiyar, Ph.D. and N. Meghanathan
Dept. of Computer Science and Software Engineering
Auburn University
Auburn, AL 36849, USA
baskiyar@eng.auburn.edu
http://www.eng.auburn.edu/~baskiyar

Keywords: Scheduling, POSIX, kernels

Received: June 26, 2004

A real-time operating system (RTOS) supports applications that must meet deadlines in addition to providing logically correct results. This paper reviews pre-requisites for an RTOS to be POSIX 1003.1b compliant and discusses memory management and scheduling in RTOS. We survey the prominent commercial and research RTOSs and outline steps in system implementation with an RTOS. We select a popular commercial RTOS within each category of real-time application and discuss its real-time features. A comparison of the commercial RTOSs is also presented. We conclude by discussing the results of the survey and suggest future research directions in the field of RTOS.

Povzetek: Podan je pregled operacijskih sistemov v realnem času.

1 Introduction

A real-time system is one whose correctness involves both the logical correctness of outputs and their timeliness [7]. It must satisfy response-time constraints or risk severe consequences including failure. Real-time systems are classified as hard, firm or soft systems. In hard real-time systems, failure to meet response-time constraints leads to system failure. Firm real-time systems have hard deadlines, but where a certain low probability of missing a deadline can be tolerated. Systems in which performance is degraded but not destroyed by failure to meet response time constraints are called soft real-time systems.

An embedded system is a specialized real-time computer system that is part of a larger system. In the past, it was designed for specialized applications, but re-configurable and programmable embedded systems are becoming popular. Some examples of embedded systems are: the microprocessor system used to control the fuel/air mixture in the carburetor of automobiles, software embedded in airplanes, missiles, industrial machines, microwave ovens, dryers, vending machines, medical equipment, and cameras.

We observe that the choice of an operating system is important in designing a real-time system. Designing a real-time system involves choice of a proper language, task partitioning and merging, and assigning priorities to manage response times. Language synchronization primitives such as *Schedule*, *Signal* and *Wait* simplify translation of design to code and also offer portability. Depending upon scheduling objectives, parallelism and communication [3] may be balanced. Merging highly cohesive parallel tasks for sequential execution may reduce overheads of context switches and inter-task communications. The designer must determine critical tasks and assign them high *priorities*. However, care must be taken to avoid starvation, which occurs when higher priority tasks are always ready to run, resulting in insufficient processor time for lower priority tasks [9]. Non-prioritized interrupts should be avoided if there is a task that cannot be preempted without causing system failure. Ideally, the interrupt handler should save the context, create a task that will service the interrupt, and return control to the operating system. Using a task to perform bulk of the interrupt service allows the service to be performed based on a priority chosen by the designer and helps preserve the priority system of the RTOS. Furthermore, good response times may require small memory footprints in resource-impoverished systems. Clearly the choice of an RTOS in the design process is important for support of *priorities, interrupts, timers, inter-task communication, synchronization*, *multiprocessing* and *memory management*.

The organization of this paper is as follows. Section 2 outlines the basic requirements of an RTOS for POSIX 1003.1b compliance. Section 3 reviews memory management and scheduling algorithms used in RTOS. Section 4, classifies popular RTOS, compares contemporary commercial RTOSs and discusses the real-time features of two popular general-purpose operating systems. Section 5 concludes by discussing the results of this survey with a few suggestions for future research.

2 Features

The desirable features of an RTOS include the ability to schedule tasks and meet deadlines, ease of incorporating external hardware, error recovery, low task switching latency, small footprint and overheads. The kernel is the core of an OS that provides task scheduling, task

dispatching and inter-task communication. In embedded systems, usually the kernel can serve as an RTOS while commercial RTOSs like those used for air-traffic control systems require all of the functionalities of a general purpose OS. In this section, basic requirements of an RTOS and POSIX compliance requirements have been addressed.

2.1 Basic Requirements

The following are the basic requirements of an RTOS:

(i) *Multi-tasking and preemptable*: To support multiple tasks in real-time applications, an RTOS must be multi-tasking and preemptable. The scheduler should be able to preempt any task in the system and give the resource to the task that needs it most. An RTOS should also handle multiple levels of interrupts to handle multiple priority levels.

(ii) *Dynamic deadline identification:* In order to achieve preemption, an RTOS should be able to dynamically identify the task with the earliest deadline. To handle deadlines, deadline information may be converted to priority levels that are used for resource allocation. Although such an approach is error prone, nonetheless it is employed for lack of a better solution.

(iii) *Predictable synchronization:* For multiple threads to communicate among themselves in a timely fashion, predictable inter-task communication and synchronization mechanisms are required. Semantic integrity as well as timeliness constitutes predictability. Predictable synchronization requires compromises [14]. Ability to lock/unlock resources is one of the ways to achieve data integrity. To illustrate this point, Java methods can be declared with the keyword synchronized, e.g. synchronized *void AddOne().* Only one thread can call a synchronized method on a particular object, other threads trying to access that method on the same object wait; thus performance degradation is possible. Molesky, Shen, and Zlokapa [12] have proposed the *Deferred Bus Theorem* for binding the waiting time on a semaphore based on the number of requesters, time spent in the critical region, and the execution times of requesting and releasing a semaphore. However, they assume that the user can estimate the time each task may hold a lock, which may not be always feasible. Although deadlines may be assigned with semaphores, there is no guarantee that critical tasks have access over non-critical tasks. Another technique achieves speedup by non-blocking (lock-free) synchronization using FIFO queues [23]. The worst-case execution time of accessing a shared data object can thus be bounded.

(iv) *Sufficient Priority Levels:* When using prioritized task scheduling, the RTOS must have a sufficient number of priority levels, for effective implementation [9]. Priority inversion occurs when a higher priority task must wait on a lower priority task to release a resource and in turn the lower priority task is waiting upon a medium priority task. Two workarounds in dealing with priority inversion, namely priority inheritance and priority ceiling protocols (PCP), need sufficient priority levels.

In a priority inheritance mechanism, a task blocking a higher priority task inherits the higher priority for the duration of the blocked task. In PCP a priority is associated with each resource which is one more than the priority of its highest priority user. The scheduler makes the priority of the accessing task equal to that of the resource. After a task releases a resource, its priority is returned to its original value. However, when a task's priority is increased to access a resource it should not have been waiting on another resource.

(v) *Predefined latencies:* The timing of system calls must be defined using the following specifications:

- *Task switching latency* or the time to save the context of a currently executing task and switch to another.
- *Interrupt latency* or the time elapsed between the execution of the last instruction of the interrupted task and the first instruction of the *interrupt handler* [4].
- *Interrupt dispatch latency* or the time to switch from the last instruction in the *interrupt handler* to the next task scheduled to run.

2.2 POSIX Compliance

IEEE Portable Operating System Interface for Computer Environments, POSIX 1003.1b provides the compliance criteria for RTOS services and is designed to allow application programmers write portable applications. The services required for compliance include the following:

- *Asynchronous I/O:* The ability to overlap application processing and application initiated I/O operations [5]. To support user-level I/O, an embedded RTOS should support the delivery of external interrupts from an I/O device to a process in a predictable and efficient manner.
- *Synchronous I/O:* The ability to assure return of the interface procedure when the I/O operation is completed [5].
- *Memory locking:* The ability to guarantee memory residence by storing sections of a process that were not recently referenced on secondary memory devices [20].
- *Semaphores:* The ability to synchronize resource access by multiple processes [17].
- *Shared memory:* The ability to map common physical space into independent process specific virtual space [5].
- *Execution scheduling:* The ability to schedule multiple tasks. Common scheduling methods include

round robin and priority based preemptive scheduling.

- *Timers:* Timers improve functionality and determinism of the system [9].
- *Inter-process Communication (IPC):* Common RTOS communication methods include mailboxes and queues.
- *Real-time files:* The ability to create and access files with deterministic performance.
- *Real-time threads*: Schedulable entities that have individual timeliness constraints [9].

3 Memory Management and Scheduling

This section addresses important issues of memory management and scheduling in an RTOS.

3.1 Memory management

An RTOS uses small memory size by including only the necessary functionality for an application while discarding the rest [22]. Below we discuss static and dynamic memory management in RTOSs.

Static memory management provides tasks with temporary data space. The system's free memory is divided into a pool of fixed sized memory blocks, which can be requested by tasks. When a task finishes using a memory block it must return it to the pool. Another way to provide temporary space for tasks is via priorities. A pool of memory is dedicated to high priority tasks and another to low priority tasks. The high-priority pool is sized to have the worst-case memory demand of the system. The low priority pool is given the remaining free memory. If the low priority tasks exhaust the low priority memory pool, they must wait for memory to be returned to the pool before further execution [1].

Dynamic memory management employs memory swapping, overlays, multiprogramming with a fixed number of tasks (MFT), multiprogramming with a variable number of tasks (MVT) and demand paging. Overlays allow programs larger than the available memory to be executed by partitioning the code and swapping them from disk to memory. In MFT, a fixed number of equalized code parts are in memory at the same time. As needed, the parts are overlaid from disk. MVT is similar to MFT except that the size of the partition depends on the needs of the program in MVT. Demand paging systems have fixed-size pages that reside in non-contiguous memory, unlike those in MFT and MVT [7]. In many embedded systems, the kernel and application programs execute in the same space i.e., there is no memory protection.

3.2 Scheduling

In this section, we very briefly outline scheduling algorithms employed in real-time operating systems. We note that predictability requires bounded operating system primitives. A feasibility analysis of the schedule may be possible in some instances. The scheduling literature is vast and the reader is referred to [15] for a detailed discussion.

Task scheduling can be either performed preemptively or non-preemptively and either statically or dynamically. For small applications, task execution times can be estimated prior to execution and the preliminary task schedules statically determined. Two common constraints in scheduling are the resource requirements and the precedence of execution of the tasks. Typical parameters associated with tasks are:

- Average execution time
- Worst case execution time
- Dispatch costs
- Arrival time
- Period (for periodic tasks).

The objective of scheduling is to minimize or maximize certain objectives. Typical objectives minimized are: schedule-length, average tardiness or laxity. Alternatively, maximizing average earliness and number of arrivals that meet deadlines can be objectives. In [15] scheduling approaches have been classified into: static table driven approach, static priority driven preemptive approach, dynamic planning based approach, dynamic best effort approach, scheduling with fault tolerance and resource reclaiming. We briefly discuss the approaches below.

(i) *Static table driven:* The feasibility and schedule are determined statically. A common example is the cyclic executive, which is also used in many large-scale dynamic real-time systems [2]. It assigns tasks to periodic time slots. Within each period, tasks are dispatched according to a table that lists the order to execute tasks. For periodic tasks, there exists a feasible schedule if and only if there is a feasible schedule within the least common multiple of the periods. A disadvantage of this approach is that a-priori knowledge of the maximum requirements of tasks in each cycle is necessary.

(ii) *Static priority driven preemptive:* The feasibility analysis is conducted statically. Tasks are dispatched dynamically based upon priorities. The most commonly used static priority driven preemptive scheduling algorithm for periodic tasks is the Rate Monotonic (RM) scheduling algorithm [8]. A periodic system must respond with an output before the next input. Therefore, the system's *response time* should be shorter than the minimum time between successive inputs. RM assigns priorities proportional to the frequency of tasks. It can schedule any set of tasks to meet deadlines if the total resource utilization less than $ln\ 2$. If it cannot find a schedule, no other fixed-priority scheduling scheme will. But it provides no support for dynamically changing task periods/priorities and priority inversion. Also, priority-inversion may occur when to enforce rate-monotonicity, a non-critical task of higher frequency of execution is

assigned a higher priority than a critical task of lower frequency of execution.

(iii) *Dynamic planning based:* The feasibility analysis is conducted dynamically—an arriving task is accepted for execution only when feasible. The feasibility analysis is also a source for schedules. The execution of a task is guaranteed by knowing its worst-case execution time and faults in the system. Tasks are dispatched to sites by brokering resources in a centralized fashion or via bids. A technique using both centralized and bidding-approach performs marginally better than any one of them but is more complex [15].

(iv) *Dynamic best effort approach***:** Here no feasibility check is performed. A best effort is made to meet deadlines and tasks may be aborted. However, the approaches of Earliest Deadline First (EDF) and Minimum Laxity First (MLF) are often optimal when there are no overloads. Research into overloaded conditions is still in its infancy. Earliest deadline first (EDF) scheduling can schedule both static and dynamic real-time systems. Feasibility analysis for EDF can be performed in $O(n^2)$ time, where n is the number of tasks [7]. Unlike EDF, MLF accounts for task execution times.

(v) *Scheduling with fault tolerance:* A primary schedule will run by the deadline if there is no failure and a secondary schedule will run by the deadline on failure. Such a technique allows graceful degradation but incurs cost of running another schedule. In hard real-time systems, worst-case blocking must be minimized for fault tolerance.

(vi) *Scheduling with resource reclaiming*: The actual task execution time may be shorter than the one determined a-priori because of conditionals or worst-case execution assumptions. The task dispatcher may try to reclaim such slacks, to the benefit of non real-time tasks or improved timeliness guarantees.

4 Commercial RTOSs

In this section, we select a prominent commercial RTOS for each class of real-time application and discuss its features. For small memory devices *Windows CE* has been discussed, for hard real-time systems, *LynxOS*, for embedded applications *VxWorks*, *Jbed* for the Java platform and *pSOS* for an object-oriented operating system. But first, we list the common capabilities of these operating systems.

- *Efficiency:* Most RTOSs are micro-kernels with low overhead. In some, almost no context switch overhead is incurred in sending a message to the system service provider.
- *Non-preemptable system calls*: Certain portions of system calls are non-preemptable to support mutual exclusion. These parts are optimized, made as deterministic as possible.
- *Prioritized scheduling*: For POSIX compliance, all RTOSs offer at least 32 priority levels. The number of priority levels range from 32-512.
- *Priority inversion control*: A means of handling priority inversion.
- *Memory management support:* Support for virtual memory management exists but not necessarily paging. The users are offered choices among multiple levels of memory protection.

4.1 Windows CE

Windows CE [13] is a modular, portable real-time embedded OS for small memory, mobile 32-bit devices. *Windows CE* slices CPU time among threads and provides 256 priority levels. To optimize performance, all threads are enabled to run in kernel mode. *Windows CE* kernel has the following features:

- While executing non-preemptive code in the kernel, translation look-aside buffer (*TLB*) misses are avoided by moving all kernel data into physical memory.
- *Kcalls*, all non-preemptable portions of the kernel, are broken into small sections reducing the duration of non-preemptable code.
- All kernel objects (such as processes, threads, critical sections, mutexes, events and semaphores) are dynamically allocated in virtual memory.
- For portability, an equipment adaptation layer isolates device dependent routines. The equipment manufacturer can specify trusted modules and processes to prevent unauthorized applications from accessing system application programming interfaces.

4.2 LynxOS

LynxOS [10] is a POSIX compatible, multithreaded OS designed for complex real-time applications that require fast, deterministic response. It is scalable from large switching systems down to small-embedded products. The micro-kernel can schedule, dispatch interrupts, and synchronize tasks. Other services offered by the kernel lightweight service modules, are TCP/IP streams, I/O and file systems, sockets, etc. In response to an interrupt, the kernel dispatches a kernel thread, which can be prioritized and scheduled similar to other threads. The priority of the interrupt handling kernel thread is the priority of the user thread that handles the interrupting device. This mechanism ensures predictable response even in the presence of heavy I/O. The OS depends upon hardware memory management units for memory protection, but does offer optional demand paging. It uses scheduling policies such as prioritized FIFO, dynamic deadline monotonic scheduling, time-slicing etc. It has 512-priority levels and supports remote operation.

4.3 VxWorks

VxWorks [21] is a widely adopted RTOS in the embedded industry with a visual development environment. It is scalable with over 1800 APIs and is available on popular CPU platforms. It offers network support, file system and I/O management. The micro-kernel supports 256 priority levels, multitasking, deterministic context switching and preemptive and round robin scheduling, semaphores and mutual exclusion with inheritance. TCP, UDP, sockets and standard Berkeley network services can all be scaled in or out of the networking stack as necessary. It can be set up so that each task has a private virtual memory upon request. For portability a Board Support Package interfaces with the hardware-dependent layer.

4.4 Jbed

Jbed [6] is a real-time operating system for embedded systems. It supports applications and device drivers written in Java. Instead of interpreting byte-codes, *Jbed* translates byte-codes to machine code prior to class loading. Its modular architecture allows dynamic code loading and scaling from small to high performance devices. It supports real-time memory allocation, exception handling and automatic object destruction. Hard real-time applications are supported via specific class libraries. It supports ten thread priority levels and EDF scheduling.

Jbed light is a smaller version for fast and precompiled applications. It contains the basic components such as the core virtual machine, a small set of standard Java libraries, and the *Jbed* libraries required to directly access peripherals. The Java virtual machine calls are implemented in the kernel. This avoids the need for a slow Java Native Interface, otherwise needed to make system calls. Current versions support ARM7, 68k and the PowerPC architectures.

4.5 pSOS

The objects, in object-oriented pSOS, include tasks, memory regions, message queues, and semaphores. It schedules tasks in preemptive priority-driven or EDF and handles priority inversion by both priority inheritance and priority-ceiling protocol. The application developer has complete control over interrupt handling. User tasks may also run in supervisory mode. Device drivers may be dynamically loaded. A memory region is a physically contiguous block of memory, created in response to a call from an application. pSOS allocates memory regions to tasks. As other objects, a memory region may be local or global.

4.6 General purpose operating systems

In this section, we outline real-time features of two popular general-purpose operating systems: Windows NT and Unix, Table 1 shows a comparison[1].

Real-time feature	Windows NT	Native Unix
Preemptive, priority-based multitasking	Yes	Yes
Deferred interrupt threads	Yes	No
Non-degrading priorities	Yes	No
Memory locks	Yes	Yes

Table 1. Real-time features of Windows NT and Unix

- *Preemption:* Although Windows NT kernel is non-preemptable there are points within the kernel where preemption is allowed. Real-time Unix also allows preemption points within system calls.
- *Deferred Procedure Calls (DPCs):* DPCs are queued calls to kernel mode functions to be executed later. They are used by drivers to schedule I/O operations that do not necessarily have to take place in an interrupt service routine at a high interrupt level and can be safely postponed until the level has been lowered. Such a mechanism allows servicing of interrupts within interrupts, if the processor disables future interrupts when an interrupt is being serviced.
- *Non-degrading priorities:* To ensure fairness, the system continuously manipulates thread priorities in Unix and Windows NT. However, Windows NT provides a band of interrupt priorities that cannot be altered. Accordingly, there exist two types of thread priorities: a real-time class and a dynamic class. Real-time class threads operate with fixed priorities that are not altered by the kernel. There are 16 priority levels in the real-time class. But any given thread is restricted only to a subset of priorities in the range of (+ or -) 2 levels of its initial priority, but not beyond the set of priorities of its class.

Although Windows NT provides fast response times, it is not as deterministic as a hard RTOS [11] because of deferred procedure calls. Since user threads have lower priority than DPCs or ISRs, mouse and keyboard handlers may preempt urgent processes. Also, DPCs are not preempted by other DPCs/threads. Furthermore, the developer has no control over third party drivers.

Since Windows NT kernel does not support priority inheritance, deadlocks may occur. It does not support prioritized queuing for inter-thread communication. In other words, if multiple threads are blocked waiting on a resource, they will be granted access in FIFO rather than priority order unlike an RTOS.

[1] Although Windows NT was not intended to be an RTOS it has been used as one in some instances.

4.7 Other commercial RTOS

Table 2 lists other common commercial RTOSs and their main features with respect to the basic requirements of an RTOS discussed in Section 2. All of the products below use a prioritized FIFO scheme for scheduling.

4.8 Research kernels

We now discuss three real-time kernels, Extensible Micro-kernel for Embedded ReAL-time Distributed Systems (EMERALDS), Spring and Arx to provide an overview of the scope and type of ongoing research in the field of RTOS. Other prominent research kernels include Chimera (from Carnegie Mellon University), Harmony (from National Research Council of Canada) and Maruti (from University of Maryland).

EMERALDS is designed for small to medium sized embedded systems [24]. It maps the kernel into every user space. Therefore a system call does not need any context switch. User level communication protocol stacks and device drivers may be added without modifying the kernel. It uses preemptive fixed priority and dynamic scheduling. A user can choose the priority of a thread based on rate-monotonic, deadline-monotonic or other fixed priority scheme. It supports 32-bit non-unique thread priorities—by setting a thread's priority to its deadline, EDF scheduling can be accomplished. The priority can be dynamically modified via a system call to support dynamic EDF scheduling. The IPC mechanisms are shared memory and message passing via mailboxes. A 32-bit priority is assigned to each message that can be used to sort them to retrieve the highest-priority message first.

Arx [16] employs user level threads for scheduling, communication and multithreading. It consists of *virtual threads* and a *scheduling event upcall* mechanism. Virtual threads provide a kernel-level execution environment for user threads. They are passive entities that are temporarily bound to user-level threads when necessary. The *scheduling event upcall* mechanism enables the kernel to notify user processes of kernel events such as thread blocking and timer expiration. User-level I/O allows programmers to write flexible and efficient device drivers for proprietary devices.

The *Spring* kernel [18] provides real-time support for distributed systems. It can schedule tasks dynamically based upon execution time and resource constraints. Thus the need to *a priori* compute the worst case blocking time for tasks is avoided. It schedules safety-critical tasks using a static table. The kernel helps retain enough application semantics to improve fault-tolerance and performance on overloads. It supports both application and system level predictability. Spring supports abstraction for process groups [19], which provides a high level of granularity and a real-time group communication mechanism. Processes within a "process group" in *Spring* work towards a common goal. *Spring* supports a system description language, which allows programmers to predefine groups and impose timing and precedence constraints on them. It supports both synchronous and asynchronous multicasting groups. It achieves predictable low-level distributed communication via globally replicated memory. It provides abstractions for reservation, planning and end-to-end timing support.

A comparison of the features of *Arx*, EMERALDS and *Spring* show that all of them incorporate most of the basic recommendations of POSIX 1003.1 b. However, the feature of real-time files has not been incorporated in any of the above research kernels.

5 Conclusion

This paper reviewed the basic requirements of an RTOS including the POSIX 1003.1b features. The POSIX 1003.1b standard does not address support for fixed-size buffers and heterogeneous multiprocessing. Designing an embedded system using an RTOS may help lower cost and the time to market. If an application has real-time requirements, an RTOS provides a deterministic framework for code development and portability. To meet the needs of commercial multimedia applications, low code size and high peripheral integration is needed. Reliability in complex real-time systems could be achieved using multilevel specifications that check the correctness of systems at compile-time and run-time.

6 References

[1] S.R. Ball, Embedded Microprocessor Systems: Real World Design, Third edition, Newnes, 2002.

[2] G. D. Carlow, "Architecture of the Space Shuttle Primary Avionics Software System," CACM, v 27, no. 9, 1984.

[3] H.Gomaa, Software Design Methods for Concurrent and Real-time Systems, First edition, Addison-Wesley, 1993.

[4] S.Heath, Embedded Systems Design, Second edition, Newnes, 2002.

[5] IEEE Information Technology—Portable Operating System Interface (POSIX)—Part1:

[6] System Application Program Interface, ANSI/IEEE Std 1003.1, 1996 Edition. Jbed RTOS, http://www.esmertec.com, Accessed Nov 15, 2004.

[7] P.A. Laplante, Real-Time Systems Design and Analysis: An Engineer's Handbook, Second edition, IEEE Press, 1997.

[8] C.L. Liu and J.W. Layland, "Scheduling Algorithms for Multiprogramming in a Hard Real-time Environment," Journal of the ACM, v. 20, no. 1, pp. 46-61, 1973.

[9] J.W.S. Liu, Real-time Systems, First edition, Prentice Hall, 2000.

[10] LynxOS, http://www.lynuxworks.com, Accessed Nov 15, 2004

[11] Microsoft Windows NT, http://www.microsoft.com, Accessed Nov 10, 2004

RTOS, Vendor	*Thread priority levels*	*Synchronization mechanisms*	*Priority inversion prevention provided*	*Development hosts, kernel characteristics*
AMX, *KADAK Products Ltd.*	N/A	Mailboxes; wait-wake requests	Yes	Windows, predictable memory block availability
C Executive, *JMI Software Systems, Inc.*	32	Messages, dynamic data queues	Yes	Windows, Solaris
CORTEX, *Australian Real-time Embedded Systems.*	62	Recursive locks, mutexes	Yes, uses priority ceiling	Windows/Unix, CPU-independent software interrupt manager; statically and dynamically segmented memory models
Delta OS, C*oreTek Systems, Inc.*	256	Semaphores, timers, message queues	Yes	Windows, Linux.
Ecos *RedHat, Inc.*	1-32	Semaphores, timers and counters	Yes, uses priority ceiling	Windows, Linux For soft real-time embedded applications in small devices
Emboss *SEGGER Microcontroller Systems.*	255	Mailbox, binary and counting semaphore	No	Windows, Linux, profiling to collect timing information for every task; task activation time independent of number of tasks.
ERTOS *JK Microsystems, Inc.*	256	Inter-thread messaging, queues, semaphores	No	Windows, DOS, OS/2. High-speed interrupt driven serial port routines
INTEGRITY *GreenHills Software, Inc.*	255	Semaphores, breakpoints can be placed any where in the system including ISRs.	Yes, mutex, semaphore	Used in critical embedded applications; object-oriented; supports distributed processing; task execution profiling.
IRIX 1.1.1.1.1 SGI	255	Message queues	Yes	SGI, Double-precision matrix support; Multi-pipe scalability
Nuclear Plus *Accelerated Technology, Inc.*	N/A	Mailboxes, pipes and queues	Yes	Windows.
OS-9 *Microware Systems Corporation.*	65535	Semaphore and queues	Yes	Windows.
OSE *OSE Systems.*	32	Message passing	Yes	Windows, Solaris, Linux. User-defined system clock resolution; fault-tolerant; suited for wireless applications.
RT-Linux *Finite State Machine Labs.*	1024	Shared memory or via files	Yes, lock free data structures and priority ceiling	Linux; supports hard real-time applications
ThreadX *Express Logic, Inc.*	32	Mutexes, counting semaphores and messaging	Yes, by disabling preemption over ranges of priorities and by priority inheritance	Windows.
QNX Neutrino *QNX Software Systems Ltd.*	64	Message passing	Yes, using priority inheritance	Windows, Solaris, Linux, Symmetrical multiprocessor systems. Every OS component runs in its own MMU-protected address space

Table 2. Features of commercial *RTOSs*

[12] L. Molesky, C. Shen, G. Zlokapa, "Predictable Synchronization Mechanisms For Multiprocessor Real-Time Systems," COINS, University of Massachusetts, *Technical Report 90-30*, 1990.
[13] C. Muench, The Windows CE Technology Tutorial: Windows Powered Solutions for the Developer, First edition, Addison Wesley, 2000.
[14] R. Ortega, "Timing Predictability in Real-Time Systems," *Technical Report*, Dept. of Computer Science, University of Washington, 1994.
[15] K. Ramamritham and J. A. Stancovic, "Scheduling Algorithms and Operating Systems Support for Real-time Systems," Proceedings of the IEEE, pp. 55-67, Jan 1994.
[16] H.Y. Seo, and J.Park. "ARX/ULTRA: A New Real-Time Kernel Architecture for Supporting User-Level Threads," Technical Report SNU-EE-TR1997-3, School of Electrical Engineering, Seoul National University, 1997.
[17] A. Silberschatz, P.B. Galvin and G. Gagne, Operating Systems Concepts, Sixth edition, John Wiley, 2001.
[18] J.A. Stankovic and K. Ramamritham, "The Spring Kernel: A New Paradigm for Hard Real-time Operating Systems," ACM Operating Systems Review, vol. 23, no. 3, pp. 54-71, 1989.
[19] M. Teo, "A Preliminary Look at Spring and POSIX 4," Spring Internal Document, 1995, available at http://www-cs.umass.edu/spring/internal/spring-kernel-docs.html
[20] The Open Group, http://www.opengroup.org/, Accessed Nov 10, 2004
[21] VxWorks, http://www.windriver.com, Accessed Nov 10, 2004
[22] C. Walls, "RTOS for Microcontroller Applications," Electronic Engineering, vol. 68, no. 831, pp. 57-61, 1996.
[23] Y. Zhang, Non-blocking Synchronization: Algorithms and Performance Evaluation, Ph.D. Thesis, Chalmers University of Technology, Sweden, 2003.
[24] K. M. Zuberi and K. G. Shin, "EMERALDS: A Small-Memory Real-Time Micro-kernel," IEEE Trans. on Software Engineering, vol. 27, no. 10, pp. 909-928, October 2001.

7 Acknowledgements

This paper has been developed while the first author was teaching classes in Real-time and Embedded Computing over the last several years. As such several students within these classes helped generously in this work to whom the first author is very grateful. The first author also thanks Dr. James H. Cross, Dept. of Computer Science and Software Engineering, Auburn University, Auburn, AL for reviewing this paper. This work was supported in part by an Auburn University internal grant and National Science Foundation Grant numbers 0408136 and 0411540.

8 Biography

S. Baskiyar is Assistant Professor in the Department of Computer Science and Software Engineering at Auburn University, Auburn, AL. His research interests are in the areas of Task Scheduling on Clusters, Computer Architecture and Embedded Systems. He has published extensively in the area of Task Scheduling on Clusters. He received the PhD and MSEE degrees from the University of Minnesota, Minneapolis and the BE (Electronics and Communications) degree from the Indian Institute of Science, Bangalore. He received the BS degree in Physics with honors and distinction in Mathematics. He received the competitive State-merit and the Indian Institute of Science scholarships. He has taught courses in Real-time and Embedded Computing, Computer Architecture, Operating Systems, Microprocessor Programming and Interfacing and VLSI Design. His experience includes working as an Assistant Professor at the Western Michigan University, as a Senior Software Engineer in the UNISYS Corporation and as an Assistant Computer Engineer in Tata Engineering and Locomotive Company Ltd., India.

N. Meghanathan received the Master's degree in Computer Science from Auburn University, Auburn, AL in 2002. He received the Bachelor's degree in Chemical Engineering from Anna University, Chennai, India. He was a research assistant in the Department of Computer Science and Software Engineering at Auburn University.

An FPGA-Based Parallel Distributed Arithmetic Implementation of the 1-D Discrete Wavelet Transform

Ali M. Al-Haj
Department of Computer Engineering,
Princess Sumaya University for Technology,
Al-Jubeiha P.O.Box 1438, Amman 11941, Jordan

Keywords: Discrete wavelet transform, Parallel distributed arithmetic, Parallel implementation, Virtex FPGAs

Received: February 7, 2004

The fine grained parallelism inherent in Field Programmable Gate Arrays (FPGAs) may be well exploited to implement the computation-intensive discrete wavelet transform, which is increasingly employed in multimedia consumer electronics. In this paper, we describe a parallel implementation of the discrete wavelet transform and its inverse using Virtex FPGAs. We make maximal utilization of the look-up table architecture of the Virtex FPGAs by reformulating the wavelet computation in accordance with the parallel distributed arithmetic algorithm. The reported single chip implementation may be used effectively in the construction of low-power, wavelet-based MPEG-4 and JPEG2000 decoders.

Povzetek: Opisana je implementacija FPGA algoritma za uporabo v uporabniški elektroniki.

1 Introduction

Digital signal processing algorithms are increasingly employed in modern wireless communications and multimedia consumer electronics, such as cellular telephones and digital cameras. Traditionally, such algorithms are implemented using programmable DSP chips for low-rate applications [1], or VLSI application specific integrated circuits (ASICs) for higher rates [2]. However, advancements in Filed Programmable Gate Arrays (FPGAs) provide a new vital option for the efficient implementation of DSP algorithms [3]. FPGAs are bit-programmable computing devices which offer ample quantities of logic and register resources that can easily be adapted to support the fine-grained parallelism of many pipelined digital signal processing algorithms [4] - [6].

At the heart of most digital signal processing algorithms is a multiply-and-accumulate function that can be implemented more efficiently with distributed arithmetic architectures [7]. These architectures make extensive use of look-up tables, which make them ideal for implementing digital signal processing functions on Xilinx FPGAs, whose architectures are based on look-up tables. Moreover, distributed architectures are suitable for low power portable applications, because they replace the costly multipliers with shifts and look-up tables [8].

An emerging arithmetic-intensive digital signal processing algorithm is the discrete wavelet transform [9] . The perfect reconstruction and lack of blocking artifacts properties of this transform have proven to be extremely useful for image and video coding applications [10]. Furthermore, the basis functions of the discrete wavelet transform match the human visual profiles, and hence provide subjectivity pleasing images at high compression rates. By virtue of such attractive features of the wavelet transform, it has been adopted by the recent multimedia compression standards MPEG-4 [11] and JPEG2000 [12].

In this paper, we describe a parallel and high speed, single-chip implementation of the discrete wavelet transform and its inverse using Virtex FPGAs [13]. We make maximal utilization of the look-up table architecture of Virtex FPGAs by reformulating the wavelet transform computation in accordance with the parallel distributed arithmetic algorithm. Unlike most papers in literature which report on single-chip VLSI architectures of the forward discrete wavelet transform only [14] - [17], this paper describes an actual implementation of both the forward and inverse transforms. Therefore, the implementation may be used in the construction of effective MPEG-4 and JPEG2000 decoders.

Finally, the paper is organized as follows. Section two gives preliminaries of the implementation which includes an overview of the discrete wavelet transform and Xilinx Virtex FPGAs. Section three describes principles of parallel distributed arithmetic, and section four describes our parallel implementation which is based on parallel distributed arithmetic. Performance results are presented in section five, and discussed in section six. Finally, some concluding remarks are presented in section seven.

2 Preliminaries

In this section we give a brief description of the Mallat algorithm which is an efficient algorithm used to

compute the coefficients of the discrete wavelet transform. We also give an overview of Xilinx Virtex FPGAs which are used as our single-chip implementation platform.

2.1 Discrete Wavelet Transform Coefficients

Wavelets are special functions which, in a form analogous to sines and cosines in Fourier analysis, are used as basal functions for representing signals. The coefficients of the discrete wavelet transform can be calculated recursively and in a straight forward manner using the well-known Mallat's pyramid algorithm [18]. Based on Mallat's algorithm, the discrete wavelet coefficients of any stage can be computed from the coefficients of the previous stage using the following iterative equations:

$$W_L(n, j) = \sum_m W_L(m, j-1) h_0(m-2n) \quad \text{.....(1)}$$

$$W_H(n, j) = \sum_m W_L(m, j-1) h_1(m-2n) \quad \text{.....(2)}$$

Where $W_L(n,j)$ is the n^{th} scaling coefficient at the j^{th} stage, $W_H(n,j)$ is the n^{th} wavelet coefficient at the j^{th} stage, and $h_0(n)$ and $h_1(n)$ are the dilation coefficients corresponding to the scaling and wavelet functions, respectively. In order to reconstruct the original data, the DWT coefficients are upsampled and passed through another set of low pass and high pass filters, which is expressed as

$$W_L(n, j) = \sum_k W_L(k, j+1) g_0(n-2k) + \sum_l W_H(l, j+1) g_1(n-2l) \quad \text{.....(3)}$$

where $g_0(n)$ and $g_1(n)$ are respectively the low-pass and high-pass synthesis filters corresponding to the mother wavelet. It is observed from Equation (3) that the j^{th} level coefficients can be obtained from the $(j+1)^{th}$ level coefficients.

Daubechies 8-tap wavelet has been chosen for this implementation. This wavelet type is known for its excellent special and spectral localities which are useful properties in image compression [19]. The filters coefficients corresponding to this wavelet type are shown in Table 1. H_0 and H_1 are the input decomposition filters and G_0 and G_1 are the output reconstruction filters.

Table 1. Daubechies 8-tap wavelet filter coefficients.

H_0	H_1	G_0	G_1
-0.0106	0.2304	-0.2304	-0.0106
-0.0329	0.7148	0.7148	0.0329
0.0308	0.6309	-0.6309	0.0308
0.1870	-0.0280	-0.0280	-0.187
-0.0280	-0.1870	0.1870	-0.0280
-0.6309	0.0308	0.0329	0.6309
0.7148	0.0329	-0.0329	0.7148
-0.2304	-0.0106	-0.0106	0.2304

2.2 Virtex FPGAs: Architecture and Programming

One of most advanced FPGA families in industry is the FPGA series produced by Xilinx [20]. The Virtex user-programmable gate array comprises two major configurable elements: configurable logic blocks (CLBs) and input/output blocks (IOBs). Each CLB is composed of two slices as shown in Figure 1. A slice contains 4-input, 1-output LUTs and two registers. Interconnections between these elements are configured by multiplexers controlled by SRAM cells programmed by a user's bitstream. The LUTs allow any function of five inputs, and two functions of four inputs, or some functions of up to nine inputs to be created within a CLB slice. This structure allows a very powerful method of implementing arbitrary, complex digital logic.

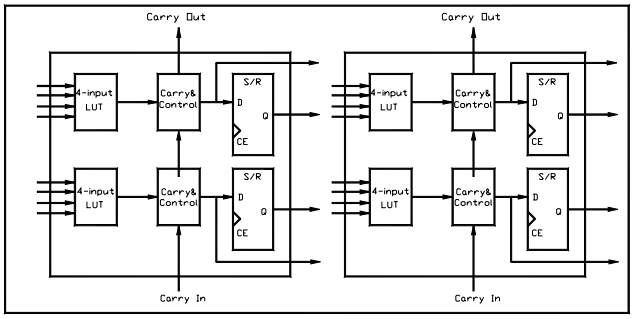


Fig. 1. Simplified Architecture of Virtex configurable logic block.

Virtex FPGAs are programmed using Verilog HDL; a popular hardware description language [21]. The language has capabilities to describe the behavioral nature of a design, the data flow of a design, a design's structural composition, delays and a waveform generation mechanism. Models written in this language can be verified using a Verilog simulator. As a programming and development environment, Xilinx ISE Foundation Series tools have been used to produce a physical implementation for the Viretx FPGA.

3 Distributed Arithmetic

Distributed arithmetic (DA) is an efficient method for computing the inner product operation which constitutes the core of the discrete wavelet transform. Mathematical derivation of distributed arithmetic is extremely simple; a mix of Boolean and ordinary algebra [22]. Let the variable Y hold the result of an inner product operation between a data vector x and a coefficient vector a. The distributed arithmetic representation the inner product operation is given as follows:

$$Y = \sum_{j=1}^{B-1}\left[\sum_{i=1}^{N} x_{ij} a_i\right] 2^{-j} + \sum_{i=1}^{N} a_i(-x_{i0})$$
$$= \sum_{j=1}^{B-1} F_j 2^{-j} - F \quad \text{.....(4)}$$

Where the input data words x_i have been represented by the 2's complement number presentation in order to bound number growth under multiplication. The variable x_{ij} is the j^{th} bit of the x_i word which is Boolean, B is the number of bits of each input data word and x_{0i} is the sign bit. Distributed arithmetic is based on the observation that the function F_j can only take 2^N different values that can be pre-computed offline and stored in a look-up table. Bit j of each data x_{ij} is then used to address this look-up table. Equation (4) clearly shows that the only three different operations required for calculating the inner product. First, a look-up to obtain the value of F_j, then addition or subtraction, and finally a division by two that can be realized by a shift.

3.1 Parallel Realization

In its most obvious and direct form, distributed arithmetic computations are bit-serial in nature, i.e., each bit of the input samples must be indexed in turn before a new output sample becomes available. When the input samples are represented with B bits of precision, B clock cycles are required to complete an inner-product calculation. A parallel realization of distributed arithmetic corresponds to allowing multiple bits to be processed in one clock cycle by duplicating the LUT and adder tree. In a 2-bit at a time parallel implementation, the odd bits are fed to one LUT and adder tree, while the even bits are simultaneously fed to an identical tree. The bits partials are left shifted to properly weight the result and added to the even partials before accumulating the aggregate. In the extreme case, all input bits can be computed in parallel and then combined in a shifting adder tree.

3.2 Virtex Implementation

The Xilinx Virtex slices have the ability to implement distributed memory instead of logic. Each 4-input LUT in a slice may be used to implement a 16x1 ROM or RAM, or the two LUTs may be combined together to create a 32x1 ROM or RAM or a 16x1 dual-port RAM. This allows each slice to trade logic resources for memory in order to maximize the resources available for a particular application. Distributed Arithmetic for inner product generation can be easily implemented in the LUT-based Xilinx Virtex FPGAs. The inner product production basically consists of table-lookup operations and additions. Thus RAM or ROM can be employed holding table values, and table lookup operations can be performed, and then a parallel adder usually follows to sum up LUT values provided by ROM or RAMs.

4 Parallel DA Implementation

The discrete wavelet transform equations can be efficiently computed using the pyramid filter bank tree shown in Figure 2. In this section we describe a parallel distributed arithmetic implementation of the filter banks shown. We start by deriving a parallel distributed arithmetic structure of a single FIR filter. We then describe the implementation of the decimator and interpolator; the basic building blocks of the forward and discrete wavelet transforms, respectively.

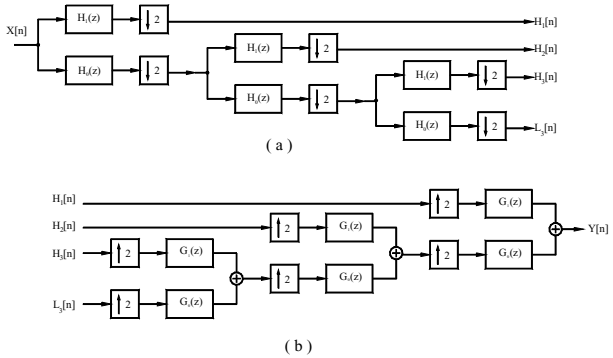


Fig. 2. Mallat's quadratic mirror filter tree used to compute the coefficients of the (a). forward and (b). inverse wavelet transforms.

4.1 Parallel DA FIR Filter Structure

All filters in the pyramid tree structure shown in Figure 2 are constructed using FIR filters because of their inherent stability. Most discrete wavelet transform implementations reported in literature employ the direct FIR structure, in which each filter tap consists of a delay element, an adder, and a multiplier [23] . However, a major drawback of this implementation is that filter throughput is inversely proportional to the number of filter taps. That is, as filter length is increased, the filter throughput is proportionately decreased. In contrast, throughput of an FIR filter constructed using distributed arithmetic is maintained regardless of the length of the filter. This feature is particularly attractive for flexible implementations of different wavelet types since each type has a different set of filer coefficients.

Distributed arithmetic implementation of the Daubechies 8-tap wavelet FIR filter consists of an LUT, a cascade of shift registers and a scaling accumulator, as shown in Figure 3. The LUT stores all possible sums of the Daubechies 8-tap wavelet coefficients given in Table 1. As the input sample is serialized, the bit-wide output is presented to the bit-serial shift register cascade,1-bit at a time. The cascade stores the input sample history in a bit-serial format and is used in forming the required inner-product computation. The bit outputs of the shift register cascade are used as address inputs to the LUT. Partial results from the LUT are summed by the scaling accumulator to form a final result at the filter output port.

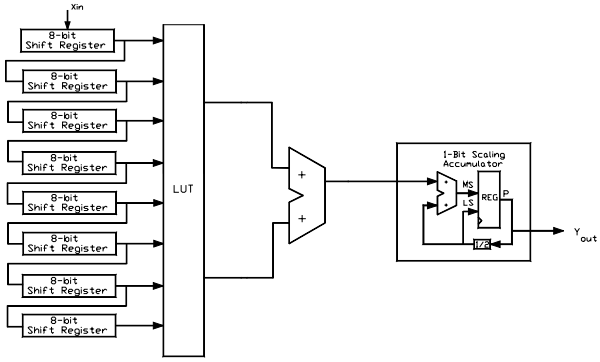


Fig. 3. A DA implementation of the Daubechies FIR filter.

Since the LUT size in a distributed arithmetic implementation increases exponentially with the number of coefficients, the LUT access time can be a bottleneck for the speed of the whole system when the LUT size becomes large. Hence we decomposed the 8-bit LUT shown in Figure 3 into two 4-bit LUTs, and added their outputs using a two-input accumulator. The 4-bit LUT partitioning is optimum in terms of logic resources utilization, since this matches naturally the Virtex slice architecture, shown in Figure 1, which uses 4-input LUTs. The modified partitioned-LUT architecture is shown in Figure 4. The total size of storage is now reduced since the accumulator occupies less logic resources than the larger 8-bit LUT. Furthermore, partitioning the larger LUT into two smaller LUTs accessed in parallel reduces access time.

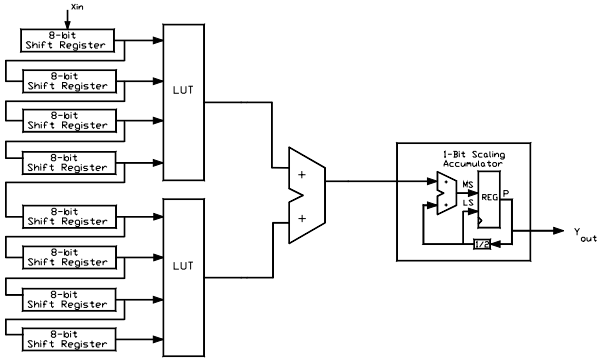


Fig. 4. A partitioned-LUT DA implementation of the Daubechies FIR filter.

A parallel implementation of the inherently serial distributed arithmetic (SDA) FIR filter, shown in Figure 4, corresponds to partitioning the input sample into ***M*** sub-samples and processing these sub-samples in parallel. Such a parallel implementation requires ***M***-times as many memory look-up tables and so comes at a cost of increased logic requirements. We describe below the implementation of our PDA FIR filter at two different degrees of parallelism; a 2-bit PDA FIR filter and a fully parallel 8-bit PDA FIR filter.

A 2-bit parallel distributed arithmetic (PDA) FIR filter implementation is shown in Figure 5. It corresponds to feeding the odd bits of the input sample to an SDA LUT adder tree, while feeding the even bits, simultaneously, to an identical tree. Compared to the serial DA filter, shown is Figure 4, the shift registers are each replaced with two similar shift registers at half the bit size. The odd bit partials are left shifted to properly weight the result and added to the even partials before accumulating the aggregate by a 1-bit scaling adder. Finally, since two bits are taken at a time, the scaling accumulator is changed from 1-to-2-bit shift (1/4) for scaling.

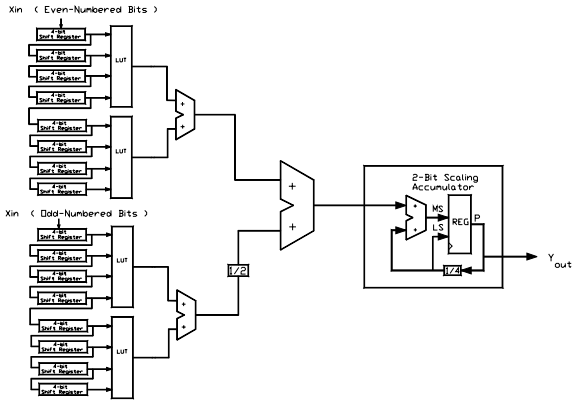


Fig. 5. A 2-bit PDA Daubechies FIR filter.

As for the fully parallel 8-bit PDA FIR filter implementation, the 8-bit input sample is partitioned into eight 1-bit sub-samples so as to achieve maximum speed. Figure 6 shows the ultimate fully parallel PDA FIR filter, where all 8 input bits are computed in parallel and then summed by a binary-tree like adder network. The lower input to each adder is scaled down by a factor of 2. No scaling accumulator is needed in this case, since the output from the adder tree is the entire sum of products.

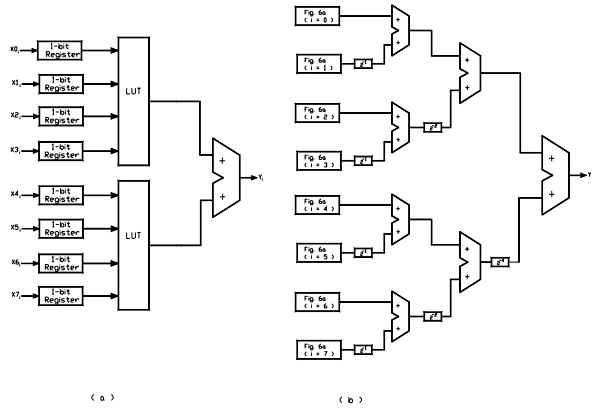


Fig. 6. (a). A single-bit and (b). an 8 -bit PDA Daubechies FIR filter.

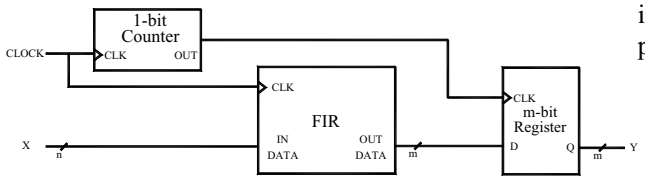
4.2 Decimator Implementation

Wavelets are The basic building block of the parallel DA forward discrete wavelet transform filter bank is the decimator, which consists of a parallel DA, anti-aliasing FIR filter, followed by a down-sampling operator [24].

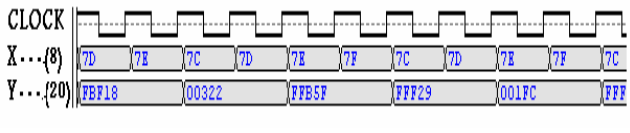
Down sampling an input sequence *x[n]* by 2 generates an output sequence *y[n]* according to the relation *y[n] = x[2n]*. All input samples with indices equal to an integer multiple of ***2*** are retained at the output, and all other samples are discarded. Therefore, the sequence *y[n]* has a sampling rate equal to half of the sampling rate of *x[n]*.

We implemented the decimator as shown in Figure 7a. The input data port of the PDA FIR filter is connected to the external input samples source, and its clock input is tied with the clock input of a 1-bit counter. Furthermore, the output data port of the PDA FIR filter is connected to the input port of a parallel-load register. The register receives or blocks data appearing on its input port depending on the status of the 1-bit counter. Assuming an unsigned 8-bit input sample is used, the decimator operates in such a way that when the counter is in the **1** state, the PDA FIR data is stored in the parallel load register, and when the counter turns to the **0** state, the PDA FIR data is discarded.

The decimator operation was modeled and verified using Verilog's functional simulator. The corresponding simulation waveform is displayed in Figure 7b. As shown, a random input sample ***X*** enters the decimator at a rate of 1sample/1 clocks , and an output filtered sample ***Y*** leaves the decimator at a rate of 1sample/ 2clocks. The input frequency is clearly halved by the decimator. We maintained sufficient precision of the decimator output sample as indicated by number of bits in the parenthesis. Allocating sufficient bits to the intermediate and output coefficients has been a necessary step to keep the perfect reconstruction capabilities of the discrete wavelet transform.



(a)



(b)

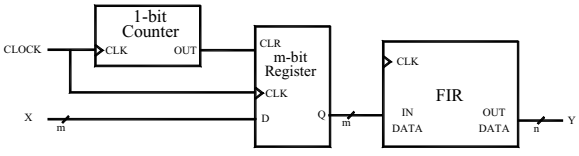
Fig. 7. (a). Implementation and (b). functional simulation of the decimator.

4.3 Interpolator implementation

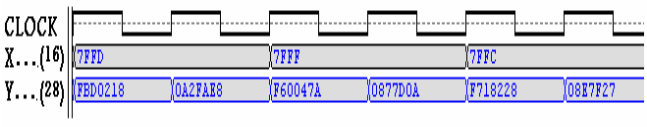
Wavelets are The basic building block of the inverse discrete wavelet transform filter bank is the interpolator which consists of a parallel DA, anti-imaging FIR filter, proceeded by an up-sampling operator [24]. In up-sampling by a factor of 2, an equidistant zero-valued sample is inserted between every two consecutive samples on the input sequence *x[n]* to develop an output sequence *y[n]*, such that *y[n] = x[n/2]* for even indices of *n,* and *0* otherwise. The sampling rate of the output sequence *y[n]* is twice as large as the sampling rate of the original sequence *x[n]*.

We implemented the interpolator as shown in Figure 8a. The input data port of the PDA FIR filter is connected to the output port of a parallel-load register. Furthermore, the input port of the register is connected to the external input sample source, and its CLK input is tied with the CLK input of a 1-bit counter. The operation of the register depends on the signal received on its active-high CLR (clear) input from the 1-bit counter. Assuming the input signal source sends out successive samples separated by 2 clock periods, the interpolating filter operates in such a way that when the counter is in the **0** state, the register passes the input sample ***X*** to the PDA FIR filter, and when the counter turns to the **1** state, the register is cleared, thus transferring a zero to the PDA FIR filter. That is, a zero is inserted between every tow successive input samples.

The interpolator operation was modeled and verified using Verilog's functional simulator. The simulation waveform is displayed in Figures 8b. The filter receives an input sample ***X*** at the rate of 1 sample/2 clocks , and sends out its filtered sample ***Y*** at the rate of 1 sample/1 clock. The input frequency is clearly doubled by the interpolator. Also, similar to the decimator, we maintained sufficient precision of the interpolator output as indicated by number of bits in the parenthesi



(a)



(b)

Fig. 8. (a). Implementation and (b). unctional simulation of the interpolator.

5 Implementation Results

We have implemented the PDA filter bank architectures described in the previous section using one of the largest available Xilinx Virtex FPGA devices, XCV300. This device contains 322,970 gates (3072slices) and can operate at a maximum clock speed of 200 MHz. Therefore, performance is usually measured with respect to two evaluation metrics; the throughput (sample rate) and is given in terms of the clock speed, and device utilization, and is given in terms number of Virtex logic slices used by the implementation.

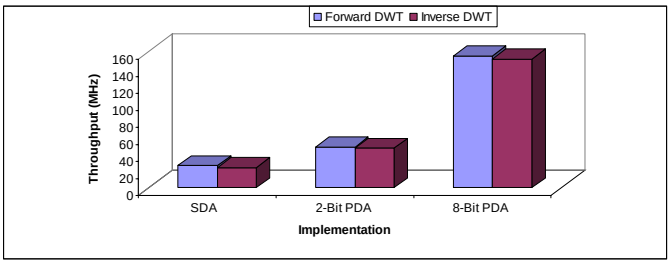
In the 2-bit PDA FIR implementation, the forward discrete wavelet transform operated at a throughput of 48.1 MHz, and required 645 Virtex slices which represents around 21 % of the total 3072 slices. Throughout of the inverse discrete wavelet transform was 46.5 MHz, and the hardware requirement was 707 slices which represent around 23 % of the total Virtex slices. On the other hand, the fully 8-bit PDA implementation, and as expected, performed much better. The forward discrete wavelet transform operated at a throughput of 154.6 MHz, and required 1167 Virtex slices which represents around 38 % of the total 3072 slices. Throughout of the inverse discrete wavelet transform was 151 MHz, and the hardware requirement was 1352 slices which represent around 44 % of the total Virtex slices.

The bit stream corresponding to the 8-bit PDA implementation was downloaded to a prototyping board called the XSV-300 FPGA Board, developed by XESS Inc [25]. The board is based on a single Xilinx XCV300 FPGA. It can accept video with up to 9-bits of resolution and output video images through a 110 MHz, 24-bit RAMDAC. Two independent banks of 512K x 16 SRAM are provided for local buffering of signals and data.

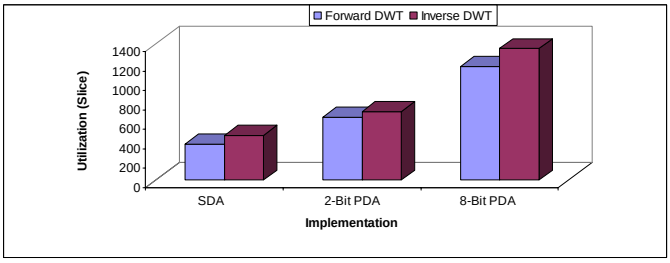
6 Discussion

In this section we compare the results presented above with the results of a serial distributed arithmetic implementation. We also compare the results of the FPGA implementations with the results of an implementation on a Texas Instruments digital signal processor. Comparison results are illustrated in Figures 9 and 10, and analyzed in the following paragraphs.

We implemented the discrete wavelet transform tree using the SDA FIR shown in Figure 4. The forward discrete wavelet transform implementation operated at a throughput of 26 MHz, and required 369 Virtex slices which represents around 12 % of the total 3072 slices. Throughout of the inverse discrete wavelet transform implementation was 23.7 MHz, and the hardware requirement was 461 slices which represent around 15 % of the total Virtex slices. It is noted from these results that there is a 6-fold performance increase for a 3-fold increase in slice count between the serial distributed arithmetic implementation and the fully parallel distributed arithmetic implementation. The results clearly demonstrate the speed/cost scalability of the distributed arithmetic algorithm, and suggest that in between the SDA and fully PDA there exist opportunities to increase performance by a factor of two or more, with corresponding increase in logic requirements.



(a)



(b)

Fig. 9. Performance comparison (a). Throughput and (b). Utilization.

The wavelet transform was also implemented on the TMS320C6711; a Texas Instrument digital signal processor with an a complex architecture suitable for image processing applications [26]. The TMS320C6711 is a highly integrated single chip processor and can operate at 150 MHz (6.7 ns clock cycle) with a peak performance of 900 MFLOPS. The processor was programmed such that the main portion of the wavelet transform was written in C, and certain sections in assembly. Also, parallel instructions were used whenever possible to exploit the abundant parallelism inherent in the wavelet transform. Sample execution times obtained for both the forward and inverse discrete wavelet transforms were 0.153 μs (6.53 MHz) and 0.276 μs (3.62 MHz), respectively.

It is noted from the results obtained above, and illustrated in Figure 10, that all distributed arithmetic FPGA implementations perform much better than the TMS20C6711 implementation. The superior performance of the FPGA-based implementations is attributed to the highly parallel, pipelined and distributed architecture of Xilinx Virtex FPGA. Moreover, it should be noted that the Virtex FPGAs offer more than high speed for many embedded applications. They offer compact implementation, low cost and low power consumption; things which can't be offered by any software implementation.

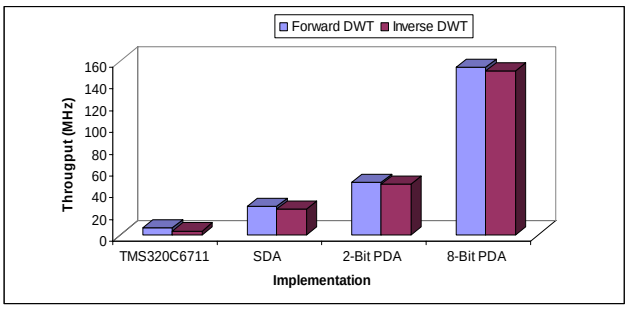


Fig. 10. Throughput performance comparison.

Finally, After completing this FPGA implementation of the discrete wavelet transform and its inverse, we are now working on integrating a whole wavelet-based image compression system on a single, dynamic, runtime reconfigurable FPGA. A typical image compression system consists of an encoder and a decoder. At the encoder side, an image is first transformed to the frequency domain using the forward discrete wavelet transform. The non-negligible wavelet coefficients are then quantized, and finally encoded using an appropriate entropy encoder. The decoder side reverses the whole encoding procedure described above. Transforming the 2-D image data can be done simply by inserting a matrix transpose module between two 1-D discrete wavelet transform modules such as those described in this paper.

7 Conclusions

In this paper we described an effective parallel single-chip implementation of the discrete wavelet transform and its inverse using Virtex FPGAs. The effectiveness of the implementation is attributed to the exploitation of the natural match which exits between the parallel distributed arithmetic technique, and the LUT-based architecture of the Virtex FPGAs. In conclusion, the implementation can be adopted in the construction of high speed MPEG-4 and JPEG2000 multimedia compression decoders.

8 References

[1] Texas Corporation, www.ti.com

[2] M. Smith, *Application-specific integrated circuits*. USA: Addison Wesley Longman, 1997.

[3] R. Seals and G. Whapshott, *Programmable Logic: PLDs and FPGAs*. UK: Macmillan, 1997.

[4] P. Kollig, B. Al-Hashimi and K. Abbot, " FPGA implementation of high performance FIR filters," In *Proc. International Symposium on Circuits and Systems*, 1997.

[5] M. Shand, " Flexible image acquisition using reconfigurable hardware," In *Proc. of the IEEE Workshop on Filed Programmable Custom Computing Machines*, Napa, Ca, Apr. 1995.

[6] J. Villasenor, B. Schoner, and C. Jones, "Video communication using rapidly reconfigurable hardware," *IEEE Transactions on Circuits and Systems for Video Technology,* vol. 5, no. 12, pp. 565-567, Dec. 1995.

[7] L. Mintzer, "The role of distributed arithmetic in FPGAs," Xilinx Corporation.

[8] K. Parhi, *VLSI digital signal processing systems*. US: John Wiley & Sons, 1999

[9] G. Strang and T. Nguyen, *Wavelets and filter banks*. MA: Wellesley-Cambridge Press, 1996.

[10] M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies, "Image coding using wavelet transform," *IEEE Trans. Image Processing*, vol. 1, no.2, pp. 205-220, April 1992.

[11] T. Ebrahimi and F. Pereira, *The MPEG-4 Book*. Prentice Hall, July 2002

[12] D. Taubman and M. Marcellin. *JPEG2000: Image compression fundamentals, standards, and practice.* Kluwer Academic Publishers, November, 2001,

[13] Xilinx Corporation. "Xilinx breaks one million-gate barrier with delivery of new Virtex series," October 1998

[14] G. Knowles, "VLSI architecture for the discrete wavelet transform," *Electron Letters*, vol. 26, no. 15, pp. 1184-1185, July 1990.

[15] A. Grzeszczak, M. Kandal, S. Panchanathan, and T. Yeap, " VLSI implementation of discrete wavelet transform," *IEEE Trans. VLSI Systems*, vol. 4, no. 4, pp. 421-433, Dec. 1996

[16] K. Parhi and T. Nishitani, VLSI architectures for discrete wavelet transforms, *IEEE Trans. VLSI Systems*, pp. 191-202, June 1993.

[17] C.Chakabarti, M. Vishwanath, and R. Owens, "Architectures for wavelet transforms: a survey," *Journal of VLSI Signal Processing*, vol. 14, no. 2, pp.171-192, Nov. 1996.

[18] S. Mallat, " A theory for multresolution signal decomposition: The wavelet representation, *IEEE Trans. Pattern Anal. And Machine Intell*., vol. 11, no. 7, pp. 674-693, July 1989.

[19] I. Daubechies, "Orthonomal bases of compactly supported wavelets," *Comm. Pure Appl. Math*, vol. 41, pp. 906-966, 1988.

[20] Xilinx Corporation. *Virtex Data Sheet*, 2000.

[21] S. Palnitkar, *Verilog HDL*, SunSoft Press, 1996.

[22] S. White, "Applications of distributed arithmetic to digital signal processing: a tutorial", In *IEEE ASSP Magazine*, pp. 4-19, July 1989.

[23] A. Oppenheim and R. Schafer, *Discrete signal processing*. New Jersy: Prentice Hall, 1999.

[24] P. Vaidyanathan, *Multirate systems and filter banks*. New Jersey: Prentice Hall, 1993.

[25] Xess Corporation. www.xess.com.

[26] Texas Instruments Corporation. *TMS320C6711 data sheet*, 2000.

JOŽEF STEFAN INSTITUTE

Jožef Stefan (1835-1893) was one of the most prominent physicists of the 19th century. Born to Slovene parents, he obtained his Ph.D. at Vienna University, where he was later Director of the Physics Institute, Vice-President of the Vienna Academy of Sciences and a member of several scientific institutions in Europe. Stefan explored many areas in hydrodynamics, optics, acoustics, electricity, magnetism and the kinetic theory of gases. Among other things, he originated the law that the total radiation from a black body is proportional to the 4th power of its absolute temperature, known as the Stefan–Boltzmann law.

The Jožef Stefan Institute (JSI) is the leading independent scientific research institution in Slovenia, covering a broad spectrum of fundamental and applied research in the fields of physics, chemistry and biochemistry, electronics and information science, nuclear science technology, energy research and environmental science.

The Jožef Stefan Institute (JSI) is a research organisation for pure and applied research in the natural sciences and technology. Both are closely interconnected in research departments composed of different task teams. Emphasis in basic research is given to the development and education of young scientists, while applied research and development serve for the transfer of advanced knowledge, contributing to the development of the national economy and society in general.

At present the Institute, with a total of about 700 staff, has 500 researchers, about 250 of whom are postgraduates, over 200 of whom have doctorates (Ph.D.), and around 150 of whom have permanent professorships or temporary teaching assignments at the Universities.

In view of its activities and status, the JSI plays the role of a national institute, complementing the role of the universities and bridging the gap between basic science and applications.

Research at the JSI includes the following major fields: physics; chemistry; electronics, informatics and computer sciences; biochemistry; ecology; reactor technology; applied mathematics. Most of the activities are more or less closely connected to information sciences, in particular computer sciences, artificial intelligence, language and speech technologies, computer-aided design, computer architectures, biocybernetics and robotics, computer automation and control, professional electronics, digital communications and networks, and applied mathematics.

The Institute is located in Ljubljana, the capital of the independent state of S**love**nia (or S♡nia). The capital today is considered a crossroad between East, West and Mediterranean Europe, offering excellent productive capabilities and solid business opportunities, with strong international connections. Ljubljana is connected to important centers such as Prague, Budapest, Vienna, Zagreb, Milan, Rome, Monaco, Nice, Bern and Munich, all within a radius of 600 km.

In the last year on the site of the Jožef Stefan Institute, the Technology park "Ljubljana" has been proposed as part of the national strategy for technological development to foster synergies between research and industry, to promote joint ventures between university bodies, research institutes and innovative industry, to act as an incubator for high-tech initiatives and to accelerate the development cycle of innovative products.

At the present time, part of the Institute is being reorganized into several high-tech units supported by and connected within the Technology park at the Jožef Stefan Institute, established as the beginning of a regional Technology park "Ljubljana". The project is being developed at a particularly historical moment, characterized by the process of state reorganisation, privatisation and private initiative. The national Technology Park will take the form of a shareholding company and will host an independent venture-capital institution.

The promoters and operational entities of the project are the Republic of Slovenia, Ministry of Science and Technology and the Jožef Stefan Institute. The framework of the operation also includes the University of Ljubljana, the National Institute of Chemistry, the Institute for Electronics and Vacuum Technology and the Institute for Materials and Construction Research among others. In addition, the project is supported by the Ministry of Economic Relations and Development, the National Chamber of Economy and the City of Ljubljana.

Jožef Stefan Institute
Jamova 39, 1000 Ljubljana, Slovenia
Tel.:+386 1 4773 900, Fax.:+386 1 219 385
Tlx.:31 296 JOSTIN SI
WWW: http://www.ijs.si
E-mail: matjaz.gams@ijs.si
Contact person for the Park: Iztok Lesjak, M.Sc.
Public relations: Natalija Polenec

INFORMATICA

AN INTERNATIONAL JOURNAL OF COMPUTING AND INFORMATICS

INVITATION, COOPERATION

Submissions and Refereeing

Please submit three copies of the manuscript with good copies of the figures and photographs to one of the editors from the Editorial Board or to the Contact Person. At least two referees outside the author's country will examine it, and they are invited to make as many remarks as possible directly on the manuscript, from typing errors to global philosophical disagreements. The chosen editor will send the author copies with remarks. If the paper is accepted, the editor will also send copies to the Contact Person. The Executive Board will inform the author that the paper has been accepted, in which case it will be published within one year of receipt of e-mails with the text in Informatica LaTeX format and figures in `.eps` format. The original figures can also be sent on separate sheets. Style and examples of papers can be obtained by e-mail from the Contact Person or from FTP or WWW (see the last page of Informatica).

Opinions, news, calls for conferences, calls for papers, etc. should be sent directly to the Contact Person.

QUESTIONNAIRE

☐ Send Informatica free of charge

☐ Yes, we subscribe

Please, complete the order form and send it to Dr. Drago Torkar, Informatica, Institut Jožef Stefan, Jamova 39, 1111 Ljubljana, Slovenia.

Since 1977, Informatica has been a major Slovenian scientific journal of computing and informatics, including telecommunications, automation and other related areas. In its 16th year (more than ten years ago) it became truly international, although it still remains connected to Central Europe. The basic aim of Informatica is to impose intellectual values (science, engineering) in a distributed organisation.

Informatica is a journal primarily covering the European computer science and informatics community - scientific and educational as well as technical, commercial and industrial. Its basic aim is to enhance communications between different European structures on the basis of equal rights and international refereeing. It publishes scientific papers accepted by at least two referees outside the author's country. In addition, it contains information about conferences, opinions, critical examinations of existing publications and news. Finally, major practical achievements and innovations in the computer and information industry are presented through commercial publications as well as through independent evaluations.

Editing and refereeing are distributed. Each editor can conduct the refereeing process by appointing two new referees or referees from the Board of Referees or Editorial Board. Referees should not be from the author's country. If new referees are appointed, their names will appear in the Refereeing Board.

Informatica is free of charge for major scientific, educational and governmental institutions. Others should subscribe (see the last page of Informatica).

ORDER FORM – INFORMATICA

Name: .....

Title and Profession (optional): .....

.....

Home Address and Telephone (optional): .....

.....

Office Address and Telephone (optional): .....

.....

E-mail Address (optional): .....

Signature and Date: .....

Informatica WWW:

large**http://ai.ijs.si/informatica/**

Referees:

Witold Abramowicz, David Abramson, Adel Adi, Kenneth Aizawa, Suad Alagić, Mohamad Alam, Dia Ali, Alan Aliu, Richard Amoroso, John Anderson, Hans-Jurgen Appelrath, Iván Araujo, Vladimir Bajič, Michel Barbeau, Grzegorz Bartoszewicz, Catriel Beeri, Daniel Beech, Fevzi Belli, Simon Beloglavec, Sondes Bennasri, Francesco Bergadano, Istvan Berkeley, Azer Bestavros, Andraž Bežek, Balaji Bharadwaj, Ralph Bisland, Jacek Blazewicz, Laszlo Boeszoermenyi, Damjan Bojadžijev, Jeff Bone, Ivan Bratko, Pavel Brazdil, Bostjan Brumen, Jerzy Brzezinski, Marian Bubak, Davide Bugali, Troy Bull, Sabin Corneliu Buraga, Leslie Burkholder, Frada Burstein, Wojciech Buszkowski, Rajkumar Bvyya, Giacomo Cabri, Netiva Caftori, Particia Carando, Robert Cattral, Jason Ceddia, Ryszard Choras, Wojciech Cellary, Wojciech Chybowski, Andrzej Ciepielewski, Vic Ciesielski, Mel Ó Cinnéide, David Cliff, Maria Cobb, Jean-Pierre Corriveau, Travis Craig, Noel Craske, Matthew Crocker, Tadeusz Czachorski, Milan Češka, Honghua Dai, Bart de Decker, Deborah Dent, Andrej Dobnikar, Sait Dogru, Peter Dolog, Georg Dorfner, Ludoslaw Drelichowski, Matija Drobnič, Maciej Drozdowski, Marek Druzdzel, Marjan Družovec, Jozo Dujmović, Pavol Ďuriš, Amnon Eden, Johann Eder, Hesham El-Rewini, Darrell Ferguson, Warren Fergusson, David Flater, Pierre Flener, Wojciech Fliegner, Vladimir A. Fomichov, Terrence Forgarty, Hans Fraaije, Stan Franklin, Violetta Galant, Hugo de Garis, Eugeniusz Gatnar, Grant Gayed, James Geller, Michael Georgiopolus, Michael Gertz, Jan Goliński, Janusz Gorski, Georg Gottlob, David Green, Herbert Groiss, Jozsef Gyorkos, Marten Haglind, Abdelwahab Hamou-Lhadj, Inman Harvey, Jaak Henno, Marjan Hericko, Henry Hexmoor, Elke Hochmueller, Jack Hodges, Doug Howe, Rod Howell, Tomáš Hruška, Don Huch, Simone Fischer-Huebner, Zbigniew Huzar, Alexey Ippa, Hannu Jaakkola, Sushil Jajodia, Ryszard Jakubowski, Piotr Jedrzejowicz, A. Milton Jenkins, Eric Johnson, Polina Jordanova, Djani Juričič, Marko Juvancic, Sabhash Kak, Li-Shan Kang, Ivan Kapustøk, Orlando Karam, Roland Kaschek, Jacek Kierzenka, Jan Kniat, Stavros Kokkotos, Fabio Kon, Kevin Korb, Gilad Koren, Andrej Krajnc, Henryk Krawczyk, Ben Kroese, Zbyszko Krolikowski, Benjamin Kuipers, Matjaž Kukar, Aarre Laakso, Sofiane Labidi, Les Labuschagne, Ivan Lah, Phil Laplante, Bud Lawson, Herbert Leitold, Ulrike Leopold-Wildburger, Timothy C. Lethbridge, Joseph Y-T. Leung, Barry Levine, Xuefeng Li, Alexander Linkevich, Raymond Lister, Doug Locke, Peter Lockeman, Vincenzo Loia, Matija Lokar, Jason Lowder, Kim Teng Lua, Ann Macintosh, Bernardo Magnini, Andrzej Małachowski, Peter Marcer, Andrzej Marciniak, Witold Marciszewski, Vladimir Marik, Jacek Martinek, Tomasz Maruszewski, Florian Matthes, Daniel Memmi, Timothy Menzies, Dieter Merkl, Zbigniew Michalewicz, Armin R. Mikler, Gautam Mitra, Roland Mittermeir, Madhav Moganti, Reinhard Moller, Tadeusz Morzy, Daniel Mossé, John Mueller, Jari Multisilta, Hari Narayanan, Jerzy Nawrocki, Rance Necaise, Elzbieta Niedzielska, Marian Niedq'zwiedziński, Jaroslav Nieplocha, Oscar Nierstrasz, Roumen Nikolov, Mark Nissen, Jerzy Nogieć, Stefano Nolfi, Franc Novak, Antoni Nowakowski, Adam Nowicki, Tadeusz Nowicki, Daniel Olejar, Hubert Österle, Wojciech Olejniczak, Jerzy Olszewski, Cherry Owen, Mieczyslaw Owoc, Tadeusz Pankowski, Jens Penberg, William C. Perkins, Warren Persons, Mitja Peruš, Fred Petry, Stephen Pike, Niki Pissinou, Aleksander Pivk, Ullin Place, Peter Planinšec, Gabika Polčicová, Gustav Pomberger, James Pomykalski, Tomas E. Potok, Dimithu Prasanna, Gary Preckshot, Dejan Rakovič, Cveta Razdevšek Pučko, Ke Qiu, Michael Quinn, Gerald Quirchmayer, Vojislav D. Radonjic, Luc de Raedt, Ewaryst Rafajlowicz, Sita Ramakrishnan, Kai Rannenberg, Wolf Rauch, Peter Rechenberg, Felix Redmill, James Edward Ries, David Robertson, Marko Robnik, Colette Rolland, Wilhelm Rossak, Ingrid Russel, A.S.M. Sajeev, Kimmo Salmenjoki, Pierangela Samarati, Bo Sanden, P. G. Sarang, Vivek Sarin, Iztok Savnik, Ichiro Satoh, Walter Schempp, Wolfgang Schreiner, Guenter Schmidt, Heinz Schmidt, Dennis Sewer, Zhongzhi Shi, Mária Smolárová, Carine Souveyet, William Spears, Hartmut Stadtler, Stanislaw Stanek, Olivero Stock, Janusz Stokłosa, Przemysław Stpiczyński, Andrej Stritar, Maciej Stroinski, Leon Strous, Ron Sun, Tomasz Szmuc, Zdzislaw Szyjewski, Jure Šilc, Metod Škarja, Jiři Šlechta, Chew Lim Tan, Zahir Tari, Jurij Tasič, Gheorge Tecuci, Piotr Teczynski, Stephanie Teufel, Ken Tindell, A Min Tjoa, Drago Torkar, Vladimir Tosic, Wieslaw Traczyk, Denis Trček, Roman Trobec, Marek Tudruj, Andrej Ule, Amjad Umar, Andrzej Urbanski, Marko Uršič, Tadeusz Usowicz, Romana Vajde Horvat, Elisabeth Valentine, Kanonkluk Vanapipat, Alexander P. Vazhenin, Jan Verschuren, Zygmunt Vetulani, Olivier de Vel, Didier Vojtisek, Valentino Vranić, Jozef Vyskoc, Eugene Wallingford, Matthew Warren, John Weckert, Michael Weiss, Tatjana Welzer, Lee White, Gerhard Widmer, Stefan Wrobel, Stanislaw Wrycza, Tatyana Yakhno, Janusz Zalewski, Damir Zazula, Yanchun Zhang, Ales Zivkovic, Zonling Zhou, Robert Zorc, Anton P. Železnikar

Informatica

An International Journal of Computing and Informatics

Archive of abstracts may be accessed at USA: http://, Europe: http://ai.ijs.si/informatica, Asia: http://www.comp.nus.edu.sg/ liuh/Informatica/index.html.

Subscription Information Informatica (ISSN 0350-5596) is published four times a year in Spring, Summer, Autumn, and Winter (4 issues per year) by the Slovene Society Informatika, Vožarski pot 12, 1000 Ljubljana, Slovenia.
The subscription rate for 2005 (Volume 29) is
– 60 EUR (80 USD) for institutions,
– 30 EUR (40 USD) for individuals, and
– 15 EUR (20 USD) for students
Claims for missing issues will be honored free of charge within six months after the publication date of the issue.

Typesetting: Borut Žnidar.
Printed by Dikplast Kregar Ivan s.p., Kotna ulica 5, 3000 Celje.

Orders for subscription may be placed by telephone or fax using any major credit card. Please call Mr. Drago Torkar, Jožef Stefan Institute: Tel (+386) 1 4773 900, Fax (+386) 1 219 385, or send checks or VISA card number or use the bank account number 900–27620–5159/4 Nova Ljubljanska Banka d.d. Slovenia (LB 50101-678-51841 for domestic subscribers only).

Informatica is published in cooperation with the following societies (and contact persons):
Robotics Society of Slovenia (Jadran Lenarčič)
Slovene Society for Pattern Recognition (Franjo Pernuš)
Slovenian Artificial Intelligence Society; Cognitive Science Society (Matjaž Gams)
Slovenian Society of Mathematicians, Physicists and Astronomers (Bojan Mohar)
Automatic Control Society of Slovenia (Borut Zupančič)
Slovenian Association of Technical and Natural Sciences / Engineering Academy of Slovenia (Igor Grabec)
ACM Slovenia (Dunja Mladenič)

Informatica is surveyed by: AI and Robotic Abstracts, AI References, ACM Computing Surveys, ACM Digital Library, Applied Science & Techn. Index, COMPENDEX*PLUS, Computer ASAP, Computer Literature Index, Cur. Cont. & Comp. & Math. Sear., Current Mathematical Publications, Cybernetica Newsletter, DBLP Computer Science Bibliography, Engineering Index, INSPEC, Linguistics and Language Behaviour Abstracts, Mathematical Reviews, MathSci, Sociological Abstracts, Uncover, Zentralblatt für Mathematik

The issuing of the Informatica journal is financially supported by the Ministry of Education, Science and Sport, Trg OF 13, 1000 Ljubljana, Slovenia.

Volume 29 Number 2 June 2005 **ISSN 0350-5596**

Informatica

An International Journal of Computing and Informatics

Introduction	N. Nedjah, L. de M. Mourelle	**123**
Investigating Strategic Inertia Using OrgSwarm	A. Brabazon, A. Silva, T. Ferra de Sousa, M. O'Neill, R. Matthews, E. Costa	**125**
Towards Improving Clustering Ants: An Adaptive Ant Clustering Algorithm	A.L. Vizine, L.N. de Castro, E.R. Hruschka, R.R. Gudwin	**143**
Efficient Pre-Processing for Large Window-Based Modular Exponentiation Using Ant Colony	N. Nedjah, L. de M. Mourelle	**155**
Max Min Ant System and Capacitated p-Medians: Extensions and Improved Solutions	F.O. de França, F.J. Von Zuben, L.N. de Castro	**163**
Application of Ant-based Template Matching for Web Documents Categorization	S.L. Ong, W.K. Lai, T.S.Y. Tai, C.H. Ooi, K.M. Hoe	**173**
Efficient and Scalable Communication in Autonomous Networking using Bio-inspired Mechanisms	F. Dressler	**183**
Model Checking Multi-Agent Systems	M. Bourahla, M. Benmohamed	**189**
End of special section / Start of normal papers		
Improving Branch Prediction Performance with a Generalized Design for Dynamic Branch Predictors	W.-M. Lin, R. Madhavaram, A.-Y. Yang	**199**
Construction of Patient Specific Virtual Models of Medical Phenomena	B. Potočnik, D. Heric, D. Zazula, B. Cigale, D. Bernad, T. Tomažič	**209**
System Resource Utilization Analysis Based on Model Checking Method	K.-S. Bang, H.-W. Jin, C. Yoo, J.-Y. Choi	**219**
Towards Neural Network Model for Insulin/Glucose in Diabetics-II	R.A. Zitar, A. Al-Jabali	**227**
A Survey of Contemporary Real-time Operating Systems	S. Baskiyar, N. Meghanathan	**233**
An FPGA-Based Parallel Distributed Arithmetic Implementation of the 1-D Discrete Wavelet Transform	A.M. Al-Haj	**241**

Informatica **29** (2005) Number 2, pp. 123–249