Volume 32 Number 3 October 2008 ISSN 0350-5596

Informatica

An International Journal of Computing and Informatics

1977

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 Higher Education, 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.

Executive Editor – Editor in Chief
Anton P. Železnikar
Volaričeva 8, Ljubljana, Slovenia
s51em@lea.hamradio.si
http://lea.hamradio.si/˜s51em/

Executive Associate Editor - Managing Editor
Matjaž Gams, Jožef Stefan Institute
Jamova 39, 1000 Ljubljana, Slovenia
Phone: +386 1 4773 900, Fax: +386 1 251 93 85
matjaz.gams@ijs.si
http://dis.ijs.si/mezi/matjaz.html

Executive Associate Editor - Deputy Managing Editor
Mitja Luštrek, Jožef Stefan Institute
mitja.lustrek@ijs.si

Executive Associate Editor - Technical Editor
Drago Torkar, Jožef Stefan Institute
Jamova 39, 1000 Ljubljana, Slovenia
Phone: +386 1 4773 900, Fax: +386 1 251 93 85
drago.torkar@ijs.si

A Role-Based Coordination Model and its Realization

Nianen Chen, Yue Yu, Shangping Ren and Mattox Beckman
Department of Computer Science,
Illinois Institute of Technology, USA
E-mail:{nchen3,yyu8,ren,beckman}@iit.edu

Keywords: coordination models, actors, roles, coordinators, open distributed and embedded systems

Received: February 1, 2008

This paper presents a framework to support Open Distributed and Embedded (ODE) application development based on the Actor-Role-Coordinator (ARC) model. The ARC model is a role-based coordination model developed to address three main concerns inherent in an ODE system: dynamicity, scalability, and stringent QoS requirements. It treats an ODE system as a composition of concurrent computation and coerced coordination. In particular, the ARC model uses concurrent objects that communicate with each other through asynchronous messages, i.e., actors, to model the concurrent computation of an ODE system, while the system's QoS requirements are mapped to coordination constraints. Coordination entities, i.e., roles and coordinators, impose coordination constraints on concurrent actors transparently through message interceptions and manipulations. In the ARC model, roles provide actor behavior abstractions for coordinators and coordinators are responsible for coordinating roles. In addition, a role also has local coordination responsibilities among actors belonging to that role. This coordination is called intra-role coordination which complements the inter-role coordination performed by the coordinators. In other words, under the ARC model, an ODE application is modeled by three orthogonal layers: computation, intra-role coordination and inter-role coordination. This separation not only improves software modularity and reusability, but also allows different levels of compositions. Our experiments show that the model scales well as the number of entities involved in the system increases, and that the performance overhead introduced by the external coordination layers is limited.

Povzetek: Opisano je ogrodje za model aktor-vloga-koordinator (ARC).

1 Introduction

Unlike most traditional software systems, open, distributed, and embedded (ODE) systems must be concerned with the environment in which they are executed. Such systems usually have rigid requirements on both the accuracy of the delivered functionality and the punctuality of its delivery. These requirements are manifested through Quality of Service (QoS) constraints, such as real-time, fault tolerance, energy consumption, and others. Another aspect of the environment is its extent. There can be many computational entities involved, and these entities are free to join or leave (intentionally or because of failures) at any time, introducing dynamicity into the system. The dynamicity and stringent QoS constraints add complexity to ODE systems, and distinguish them from traditional concurrent distributed systems.

Concurrent distributed computation models have been well studied over the past decades. CSP [21], π-calculus [33], and the actor model [1, 2] are good examples. These models are still widely used today as they provide a uniform way to model diversified applications. For instance, the Actor model treats "actors" as universal primitives: in response to a message an actor receives, the actor may make local decisions and decide how to respond to the next message received, create more actors, and/or send more messages. It is often used as a framework for modeling, understanding, and reasoning about a wide range of modern concurrent systems. For instance, Web Services with SOAP endpoints can be modeled as actors [20, 19]; an agent-based system can be modeled as an actor system, where (mobile) agents are modeled as (mobile) actors [28, 24]; and Sensor and Actor Network (SAN) is recently proposed to use the Actor model as the theoretical basis for sensor networks [26, 12, 7].

However, these models are well-defined mathematical abstractions for concurrent computation in an ideal distributed environment, in which simplifying assumptions are made to reduce the complexity of the models. For instance, communication among distributed entities is assumed to be both reliable and instantaneous. The focus of these models is on the functional behaviors of the computation. This may suffice for traditional and general purpose concurrent distributed applications, but for ODE systems, such assumptions about the run-time environment often do not hold. For example, in most embedded applications, a message that does not arrive on time is considered a fault, but traditional distributed computation models do not make any guarantees about such QoS promises. What we need is a model to study QoS aware interaction, or coordination, among distributed computational entities in ODE systems. This model should accurately exhibit an ODE application's functional behaviors, and also precisely reflect the application's context, taking into account the dynamicity and stringent QoS requirements.

In order to conquer the complexity and dynamicity inherent to ODE systems, we may decompose these systems into different concerns. Separation of concerns as a software engineering principle is not new [18, 3]. However, how a concern is delineated plays a critical role in the quality of the delivered software models. A concern should be logically self-contained and, ideally, orthogonal and transparent to the other concerns in order to minimize the interference among them.

For instance, an open embedded real-time application, such as an environmental monitoring system, will send data from wide-area sensors to data processing entities on the Internet. The results are fed back into the physical world for actuation. In order to interact with the physical world in real time, open embedded applications must be able to fulfill a fundamental requirement, that fresh data be available at the right computation site at the right time. However, as Kang et. al [23] pointed out, current computing and communication-oriented paradigms face a huge obstacle in achieving this vision of open embedded real-time systems. Therefore, instead of interacting directly with a number of distributed data sources or actuators, it is important to have high level abstractions that federate distributed entities, coordinating them to abide by QoS requirements.

Consider the following simplified scenario as an example of the problem our research addresses. Suppose we have deployed infrared and radio wave sensors in an open space to detecting foreign objects. As shown in Figure 1, depending on the exact location of the foreign object, different groups of sensors will be active and generate data. In order for a control center to take appropriate action, data from the two types of sensors must be semantically consistent (i.e., indicating the same type of object) and they must arrive at the center within a specified time range.

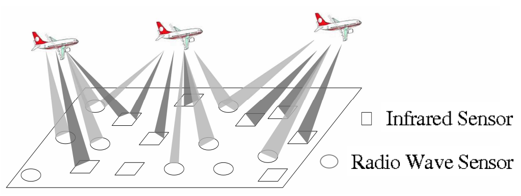


Figure 1: Open Space Surveillance

Clearly, it is a *must* that the infrared and radio wave sensors be *coordinated* in a *timed* fashion, but the nature of the problem prohibits us from statically pairing them up. The key technical challenge is that coordination is necessary, and that coordination itself is subject to QoS constraints. Furthermore, the coordinatees constitute a large and dynamically changing set. Integrating the coordination requirements into the basic computation description is not a viable solution; it only complicates an already hard problem. Unfortunately, existing research has not approached ODE applications from the coordination angle, neither have earlier coordination models addressed coordination under QoS constraints in depth. Therefore, new research is needed to support the development of ODE applications.

In this paper, we present a framework for developing ODE applications based on a role-based distributed coordination model, the Actor, Role and Coordinator (ARC) [44] model. The focus of this ARC model is to separate the QoS or non-functional requirements from the embedded applications' functional logic, and at the same time to address the dynamicity and scalability issues inherent to ODE systems. In particular, the actor layer models the concurrent computational part of an ODE system, while an independent coordination model is developed to address the federation of distributed entities to satisfy the system's QoS requirements. The coordination model contains both the coordinator layer and the role layer; the role layer provides a level of abstraction to mask the dynamicity of the actor layer from the coordinators, and each role coordinates the local group of actors that share that role. This further reduces the complexity of coordinators and improves coordination scalability. We present in detail a CORBA based implementation of ARC that provides architectural support for transparent application of QoS constraints on concurrent computations. The design criteria of the framework are performance, scalability, and flexibility.

The rest of the paper is organized as follows: Section 2 discusses related work. Section 3 presents the ARC model and the composition of three autonomous entities, i.e., actors, roles, and coordinators. Section 4 presents an ARC framework and preliminary evaluation results. Finally, we conclude in Section 5.

2 Related work

Recent research has yielded significant results on coordination models and languages. In their landmark survey [41], Papadopoulos et. al. conclude that coordination models can be classified into two categories, namely data-driven and control-driven. The tuple space model (Linda) [10] represents the data-driven category, and has been extended with such systems as Lime[42], Klaim[36], and related extensions [37]. Systems such as the Ideal Worker Ideal Manager (IWIM) model [4] presents a control-driven or "exogenous" category. Recently, tuple center and ReSpecT [40, 38] provide a hybrid view.

Control-driven models, such as Abstract Behavior Types (ABT) [5], Law Governed Interaction (LGI) [34], ROAD [11], Reo [6], Orc [35], and CoLaS [13] isolate coordination by considering functional entities as black boxes. For example, the ABT model extends the IWIM model by treating both computation and coordination components as composable Abstract Behavior Types. Like IWIM, ABT is a two-level control-driven coordination model where computation and coordination concerns are achieved in separate and independent levels. The Reo model uses a circuit-like network of composable channels to provide communication between components. Components send messages across these channels, and the geometry of the channels determine the destination or destinations of the messages. The Orc model uses "site calls" to model computation [43]. Unlike Reo channels, Orc's site calls are not expected to be persistent.

The concept of role is seen in object-oriented systems when a set of common behaviors is abstracted and can be assigned to an object [15, 25]. Roles are an important technique in a variety of computing systems. For example, in the computer security area, the Role Based Access Control (RBAC) [14] model uses roles to separate users from security policies in order to achieve scalability and flexibility. In object-oriented programming [27] and in design patterns [17], roles are used to represent solutions and experiences. There are control-driven models, such as ROAD, CoLaS, TuCSoN with Agent Coordination Contexts (ACC) [39] and Finesse [8], to name a few, that try to mitigate the scalability issues of open distributed systems by adopting role concepts. Most current role-based coordination models are based on organizational concepts, where roles abstract coordination behaviors among participants who play the roles. Cabri presents a survey of role-based coordination models in [9]. Additionally, quite a few coordination models take decentralization into account. TuCSoN [40] distributes communication abstractions (tuple centers) to Internet nodes. Every tuple center produces and maintains its own local coordination rules. CoLaS divides the whole distributed system into multiple coordination groups. Each coordination group takes care of an independent set of coordination policies. ROAD provides a recursive structure that composes fine-grained, small coordination groups into coarse-grained, large ones. LGI follows a controller metaphor and provides a controller for every object in the system, and hence implements a full-fledged decentralization.

The ARC [44] model differs from these models by separating inter-role coordination and intra-role coordination and distributing the coordination activities to coordinators and roles respectively. Roles are active entities with coordination ability instead of merely abstract interfaces. The distribution of coordination responsibility is based on the functionalities of the roles and is therefore more logical and customizable. The emphasis on active roles and the corresponding separation of inter-role and intra-role coordination distinguishes the ARC model from previous role-based coordination models.

A similar actor oriented model is advocated by Lee et al. [30, 31]. In this model, actor executions and communications are under the guidance of a "model of computation," which gives operational rules to determine when and how actors can perform their computations, update their states, or send messages to other actors. The model of computation separates the communication mechanisms and work flows of actors from their computational designs, such that reusability is possible and compositions of components are more robust.

Though the above actor-based models, like ARC, separate coordination from the functional core of a system based on concurrent actors, the focus of ARC is to address the dynamicity and scalability issues in coordinating large set of autonomous and asynchronous entities. The emphasis on role-based coordination distinguishes the ARC model from previous multi-level actor-based coordination architectures.

A set of coordination models has been proposed to address the coordination issues based on the Actor model [1, 2], such as Frølund's *Synchronizer* [16], Venkatasubramaniam's *TLAM (Two-Level-Actor-Model)* [50], and Varela's *director* [49]. One common theme of these models is the use of reflection with actors. This can be seen is systems such as ActorNet [29], and Reflective Russian Dolls (RRD) [32]. ActorNet provides a platform designed for small, heterogeneous systems. It provides a uniform environment for the actors, and makes use of `call/cc` to allow actors to migrate themselves to other nodes in the system. RRD is similar to ARC in that there are levels of coordination. Both achieve coordination by using reflection to modify the delivery of messages.

ARC, however, is a three-layer system, with functional behavior confined to the lowest level, and coordination to the upper two levels. The formal semantics of the ARC model is given in [44]. The RRD is a multi-level system; each level encapsulating the levels below it. The formal comparison between the ARC model two other coordination models, i.e. the and Reflective Russian Dolls (RRD) [47] and Reo [6], is given in [46]. Yu and et al. used Maude to further verify safety properties that can be imposed through the ARC model [51].

3 The Actor, Role, and Coordinator (ARC) model

In this section, we discuss in detail the Actor-Role-Coordinator model.

3.1 The actor model

We use active objects, i.e., actors [1, 2], to model asynchronous and distributed computations. The choice of the actor model as a foundation for the underlying computations of an ODE system is in many ways a natural one. The actor model is inherently concurrent, and systems of actors are open and distributed. However, the basic actor model does not enable the coordination of groups of actors to be specified in a modular fashion. This greatly limits their usefulness in the ODE domain. The ARC model eliminates this impediment by introducing exogenous coordination objects, i.e., roles and coordinators.

Actors are autonomous, active entities that communicate with each other through asynchronous messages. Each actor has a unique mail address and a mailbox to receive messages. Unprocessed messages are buffered at the receiving actor's mailbox. Within each actor, there is a single thread of control that processes messages sequentially. Each actor has its own states and state dependent behaviors. The states are encapsulated and can only be changed by the actors themselves while processing messages. Different actor states may decide different behaviors that in turn affect how messages are processed. While processing a message, an actor may perform three primitive operations: send asynchronous messages to other actors, create new actors, or change its own states (become) and then become ready to retrieve the next available message in the mailbox. Figure 2 pictures the internal structure of the actors.

Here is a simple example to demonstrate the actor model. Assume an operation can be performed by a computational entity (namely, an actor) called an "executor" once and only once. Any actor that is not an executor is called a "forwarder." We distinguish an executor from a forwarder by looking at its internal state *executed*. If *executed* is false, then this actor is an executor, otherwise it is a forwarder. The behavior of an executor is as follows: when it receives a message requesting the service, it performs the service, and sets its state *executed* to be true, which triggers the actor to become a forwarder. Finally, the former executor creates another actor with the same behavior (i.e., perform the same operation) and with its state *executed* set to be false. In other words, this actor becomes a forwarder and creates a new executor. After becoming a forwarder, this actor changes its behavior. When the same message arrives at the forwarder, instead of executing the operation, it forwards the message to the executor that it created. This behavior is recursive; the "executor" to which it forwards the message may also have become a forwarder, and will in turn continue forwarding the message to successive forwarders until the current executor is located. This example explains the basic concepts of an actor and its three primitives: *send*, *create*, and *become*. All actor based computations can be implemented by these three primitives.

3.2 The abstraction levels of ARC

In the ARC model, a role is a static abstraction for behaviors shared by a set of underlying computational actors. This abstraction decouples behaviors from their implementors and eliminates static binding among computational actors. It also shares coordination responsibilities. More specifically, there are two types of *active* coordination objects in the model: *roles* and *coordinators*. The coordination is partitioned into intra-role and inter-role coordinations and distributed among roles and coordinators, respectively. The coordinators (i.e., inter-role coordination objects) coordinate behaviors while the roles (i.e.,intra-role

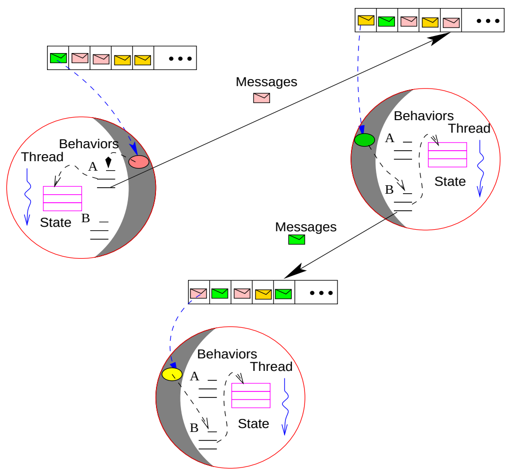


Figure 2: The Actors

coordination objects) coordinate members that share the same behavior.

Coordinators constrain the coordination behavior of roles. This eventually affects a message's dispatch time and location (target) in a computation. However, computational actors and coordinators are transparent to each other. Hence, the dynamicity inherent in the computation is hidden from the coordinators. Compared to the number of actors involved in an ODE application, the number of behaviors (and therefore the number of roles) contributed by these actors is usually order(s) of magnitude smaller. Therefore, the model is not only stable, but also scalable.

Under the ARC model, the open space surveillance system introduced in Section 1 (Figure 1) can be mapped to a set of sensor *actors*, two *roles* for the infrared sensors and radio wave sensors, respectively, and a *coordinator* (Figure 4). The inter-role constraint is on the time relation of the data coming from the infrared sensor role and the radio wave sensor role. Each role can have different intra-role coordination policies. For instance, the infrared role may ensure synchrony by waiting for data from all its members, while the radio wave role only waits for data from a majority of its members.

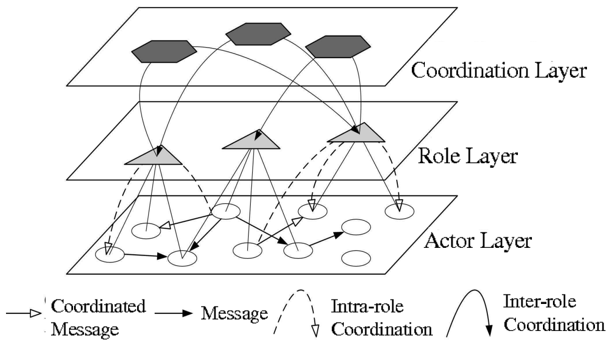


Figure 3: The ARC Model

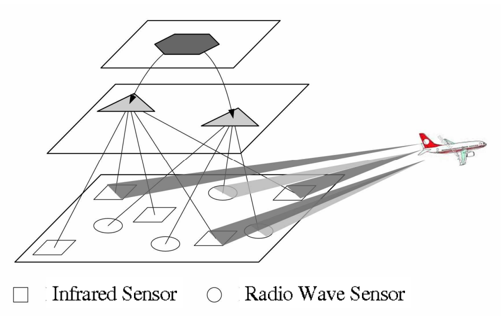


Figure 4: The ARC View of an Open Space Surveillance System

The separation of *computation*, *intra-role*, and *inter-role coordination* advocated by the ARC model is clean and orthogonal. This separation mitigates the complexity of each individual type — coordinators only concern themselves with coordinating a *small scale* of roles while roles care *only* about actors of the *same behavior*. This provides grounds for independent modeling and compositional reasoning.

Separation and transparency are the results of the following properties of the ARC model:

1. The actor layer does not depend on the coordination layer. The actors fulfill their functional behaviors independently by exchanging messages without any knowledge that the coordination entities even exist.

2. The coordination layer intercepts messages among actors and applies coordination constraints on the messages. Coordination does not require direct message interactions between actors and coordination entities.

Computation actors carry out their logical computations by reacting to messages received. As a result, if the roles or coordinators do not send any computational messages to the computation actors, the underlying computation will retain its computational properties.

The role layer bridges the actor and the coordinator layers and may therefore be viewed from two perspectives. From the perspective of a coordinator, a role enables the coordination of a set of actors that share the same static description of behaviors without requiring the coordinator to be aware of the individual actors in the set. From the perspective of an actor, the role is a coordinator that actively imposes coordination constraints on messages sent and received by the actor.

Though actors, roles, and coordinators have different responsibilities, we uniformly model their behaviors using actors. To comply with the separation of concern principle, we categorize these actors into two types: computation actors that capture system computation concerns, and coordination actors that abstract system coordination concerns. More specifically, roles and coordinators are coordination actors, whereas actors in the actor layer are computation actors. Thus, coordination actors are actors which satisfy the basic actor semantics by providing the actor operational primitives. However, they are special actors that are able to handle specific types of messages, namely, events.

In our model, actors communicate with each other via messages, which are defined as a three-tuple $< rcver - actor, op, par >$. Here $rcver - actor$ is the name of the recipient actor, op is the behavior name that the recipient actor is required to apply, and par contains the parameters that the recipient needs to perform its behavior.

Events are special messages that are atomically dispatched on coordination actors. Unlike computation messages, the recipient of an event is not an individual coordination actor, instead, events are broadcast to all coordination actors in the system. Thus, an event is defined as $< All, op, par >$ where All indicates that the event is broadcast. Though events are broadcast to all roles, we may instead use an intermediate "default" role as a mediator to receive and forward events between actors. This optimization can convert the broadcast into a two-element group-cast, reducing the communication overhead and synchronization complexity.

Another important characteristic of events is that an event is instantaneous and atomic. In other words, the generation of an event and the consumption of an event are atomic, and no actor computation messages can be processed during this period of time. This requirement guarantees that coordination constraints are applied on related messages before these messages are dispatched on computation actors.

To maintain coordination transparency and avoid interfering with the computation actors' functionalities, coordination actors are not allowed to generate or send messages to computation actors. The computation level and coordination level are connected through *events*. While messages are used between computation actors to carry out computations, events represent state changes in the system and trigger coordination related behaviors on coordination actors.

There are three events defined in an actor layer: *send* (a message is sent by an actor), *new* (creation of a new actor), and *ready* (change actor behavior if necessary and ready for next available message). All these events from computation actors are observable by roles. Upon observing the events, the roles cooperate with coordinators through *inter-role* and *intra-role* constraints to coordinate *when* and *where* messages should be dispatched among computation actors.

3.3 Roles and their responsibilities

Since an ODE system may have a large number of computational entities that are free to join or leave autonomously, the underlying actors modeling them could also be both large in number and very dynamic. Basing the stability and scalability of coordination policies on the actors themselves will be difficult. In an ODE system, however, the set of well-defined *behaviors* is limited and less dynamic. Therefore we introduce roles as a means of representing abstractions for these system behaviors; this enables us to conceal the dynamicity and scale of the ODE environment.

In addition to representing abstractions for the properties of the system behaviors, roles also are responsible for actively coordinating their players to achieve coordination requirements. Roles serve as an abstraction by specifying membership criteria, i.e., a static specification of functional behaviors that computation actors belonging to the role must have. The role is responsible for managing the integrity of its membership. Roles also actively coordinate their member actors in order to satisfy coordination requirements. The *intra-role* coordination coerced by roles realizes and complements the *inter-role* coordination enacted by coordinators.

Membership management behavior Before a role can perform its membership management activities, the behavior abstraction, i.e. the role membership criteria, must be specified. We use logic expressions of actor states and operations to describe the criteria. More specifically, the role membership criteria are represented by a tuple $< O, A >$, where O is a set of message types (operations) that an actor must be able to process, and A is a set of attributes that actors need to display for joining the role. Any actor that is controllable by coordination rules must declare its own functional behavior, using the same tuple format.

Upon observing a *new* event or a *ready* event from a computation actor, the role acquires the newly updated behavior from the computation actor and compares it against its membership criteria. It then determines whether the actor should be added to the membership list (the actor behavior matches the role criteria), ignored (the actor was not a member and its behavior does not match the role criteria), or removed from the role (the actor was a member but its new behavior does not match the role criteria). More precisely, a role's management behavior is a mapping from a set of actor events to membership updates. Note that according to the semantics of the actor model, actors are free to reject exposing their internal states to the roles. This allows an actor to reject coordination. Such actors will belong to a "default" role that performs no coordination.

Each role has a distinct purpose. This requirement disallows overlapping criteria among roles, eliminating the possibility that conflicting constraints will be imposed on an actor by multiple roles simultaneously. This requirement has its basis in the underlying actor model: each actor has only a single thread of control and therefore may play only one role at any given time. More precisely, let $C(\gamma)$ denote the role membership criteria declared by role γ, and let $B(\alpha)$ denote the functional behaviors provided by an actor α. As we have discussed, the actor functional behaviors and the membership criteria are both represented as a comparable tuple $< O, A >$. To be added to a role, the actor functional behaviors have to match the role's membership criteria. A and Γ denote the set of actors and roles in the system, respectively, and $F : A \rightarrow \Gamma$ is a function that assigns an actor to a role. At any given time, well-defined roles and actors in a system must satisfy the following requirements:

1. Roles are exclusive: role declared behaviors do not overlap, i.e.,
$$\forall \gamma, \gamma' \in \Gamma : C(\gamma) \cap C(\gamma') = \phi$$
2. Roles are exhaustive: every actor belongs to one of the roles, i.e.,
$$(\bigcup_{i}^{n} B(\alpha_i) = \bigcup_{j}^{m} C(\gamma_j)), \text{ and}$$
$$(\forall \alpha \in A, \exists \gamma \in \Gamma : B(\alpha) = C(\gamma))$$
3. Roles are repetitive: repeated actor behaviors replicate the assignment of the actor to the same role, i.e.,
$$\forall \alpha_i, \alpha_j \in A : B(\alpha_i) = B(\alpha_j) \Rightarrow F(\alpha_i) = F(\alpha_j)$$
4. Each actor only plays one role at a given time, i.e.,
$$\forall i, j,\ j \neq i : B(\alpha) = C(\gamma_i) \Rightarrow B(\alpha) \neq C(\gamma_j)$$

Coordination Behavior As roles are abstractions of functional behaviors, it is possible that more than one actor may belong to a specific role at any given time. Actors playing the same role may need to coordinate with each other to satisfy certain QoS constraints. Such constraints are called *intra-role* coordination constraints.

A role's coordination behavior thus has two aspects: (1) it retrieves *inter-role* constraints specified by the coordinators; (2) it is responsible for enforcing both the *inter-role* and *intra-role* coordination constraints on actors. Since roles are coordination actors and are not allowed to send/receive messages to/from computation actors, message interception and manipulation is the only feasible means to apply the constraints. Furthermore, all these behaviors are triggered by observed events on computation actors. Therefore, the coordination behavior of a role can be given the following interpretation: upon observing an event from a computation actor, and based on its current states, the role may manipulate messages, generate events (which are observable by coordinators), or change its own states.

The coordination rules are enforced on actors without their awareness. The involvement of roles in the coordination process causes coordination in the ARC model to be decentralized. Active roles cause our coordination model itself to become a distributed subsystem, inheriting the full benefits that a distributed system may offer.

We can use a "Video on Demand" (VoD) application as an example to depict a role's behaviors. We assume there are multiple VoD client actors and VoD server actors in a distributed environment. Each VoD client actor can perform a *request_video* operation, while the server actor can perform *send_video* operation. However, clients may have different requirements, which need to be met by receiving different services from servers. We therefore separate the client actors into different roles depending on their level of service attributes, i.e. a VOD client actor has a *Regular_VOD_Client Role* if its *level_of_service* attribute is set to *regular*; while a VoD client has a *VIP_VoD_Client* role if its *level_of_service* attribute is set to *very_important*. Therefore, when an actor is created or moves into the system, the roles will check its operations and attributes. For example, if a *VIP_VoD_Client* role finds that the new actor has a behavior tuple <<*request_video*>, <*level_of_service:regular*>>, which matches its role criteria, it will then help the actor join its group by performing its member management behavior.

To explain the role's coordination behavior, we assume that there are multiple VoD server actors in the environment, each of which has different resources (CPU speed, workload, memory, reserved network throughput, and so on). Based on the requests from different types of VoD clients, the *VoD_Server* role decides which server actor shall be assigned to process the current request. For example, if the request is from an actor with a *VIP_VoD_Client* role, this request will be forwarded to a *VoD_Server* actor with the highest available resources. This coordination rule is applied on the actors directly within a role, but not among roles, therefore it belongs to *intra-role* coordination.

3.4 Inter-role coordination — coordinators

In contrast to *intra-role* coordination, coordination among high-level coarse-grained roles are called *inter-role* coordination. We define another type of coordination actor, the *coordinator*, to specify *inter-role* coordination policies. These policies are written in terms of roles. A policy is a set of constraints over a set of properties. Values associated with a property are drawn from an enumerable domain. A constraint specifies a boolean relation involving a set of properties.

Similar to roles, coordinators are also active objects and impose coordination constraints based on their states. However, in our model the actor layer and the coordinator layer are mutually transparent. Coordinators do not directly apply coordination constraints on computation actors, neither do actors know of the existence of coordinators. Coordinators specify and impose policies based on abstract actor functionalities, but not on individual actors.

The role layer bridges the coordinator and actor layers. Roles propagate the events observed from the actors to the coordinators. Upon receiving such events, the coordinator locates constraints in its constraint store based on current states, and propagates the constraints to roles where these constraints are imposed on computation actors.

Consider an example in which multiple producers and multiple consumers share the same buffer. We use a producer role and a consumer role to capture the producers and consumers, respectively. The two roles must coordinate to respect the causal order (an item must be produced before it can be consumed) and buffer size. Instead of specifying the coordination among each pair of producer and consumer, we impose the coordination upon the roles which will in turn propagate the constraints to the role players.

3.5 Composition of concurrent computation and coerced coordination

Based on the ARC model, an ODE system can be specified in three steps. First, establishing the underlying functional computations (modeled by computation actors). Second, implementing the computational actors to carry out the computation. Finally, embedding the functional objects in an environment constrained by coordination actors. Here we focus on QoS constraints that can be achieved by manipulating the messages in the time and actor space dimensions. Example manipulations on the time-axis include moving messages to the beginning of the actor's mail queue, blocking them, or postponing them to later time. Manipulations on the actor space domain include taking messages sent to one particular actor and duplicating, rerouting, or broadcasting them to other actors to satisfy fault tolerance, security and other QoS requirements.

Coordination actors observe events occurring at the computation actor layer, and perform coordination behaviors accordingly. However, coordination actors are partitioned into roles and coordinators, and these two types of coordination actors also need to collaborate with each other. Their collaborations are achieved through event exchanges. The events that are observable in the ARC model are presented in Table 1.

Note that the events specified in Table 1 do not exactly follow those defined in the traditional actor model [1, 2]. In Agha's actor model, there are only three primitive events, *send*, *new*, and *ready*, where *ready* actually represents two behaviors of an actor: become a new actor and ready for next available message. After processing a message, even if an actor does not change its behavior, it still has to perform *become* to become itself. However, in the ARC model the change of behavior triggers the roles' membership management behaviors. If we perform *become* each time a message is finished processing, we will continuously trigger the member management actions in roles, which in most cases will be unnecessary. For this reason, in the ARC model we separate the *ready* event into two events, namely the *become* event and *ready* for next available message event, where *become* explicitly specifies that an actor changes its behavior and triggers membership management actions.

After a message has been sent out to a recipient computation actor, and before it can be processed, a $send(msg)$ event is broadcast and needs to be handled by the coordination actors. The argument msg is the message that has been sent. After processing the current message, the actor will enter a state in which it is ready to process the next available message in the mail queue. This will cause a $ready(msg)$ event to be broadcast to trigger coordination behaviors. Events are instantaneous; coordination actors

Location	Event	Triggered By
Actor	$send(msg)$	A computation actor performs a *send(msg)* operation.
	$new(beh)$	A computation actor performs a *create(beh)* operation.
	$become(beh)$	A computation actor performs a $become(beh)$ operation, where beh represents a behavior that is different from the actor's current behavior.
	$ready(msg)$	A new message in the actor's mailbox is dispatched at the actor.
Role	$propSend(msg)$	A $send(msg)$ event from a computation actor is observed.
	$propReady(msg)$	A $ready(msg)$ event from a computation actor is observed.
Coordinator	$tell(inter-roleconstraints)$	A $propSend()$ or $propReady()$ event is observed.

Table 1: Events Observable in the ARC Model

observe and handle events atomically. Message deliveries, on the other hand, always take time. The dispatch of a message will always happen at a later time than when the message was sent. Therefore, it is guaranteed that coordination actors can perform their coordination behaviors on messages in the recipient actors' mailboxes before those messages are processed.

The $new(beh)$ or $become(beh)$ events are triggered when a new actor is created or when an actor changes its behavior. The argument beh is the behavior of the new actor to be created, or the new behavior an actor obtains. All roles in the system are able to observe such an event and compare the behavior with their membership criteria. The role whose membership criteria matches the computation actor's behavior adds the computation actor into its group. For completeness of the roles in our system, we also introduce a *default* role. If the actor's behavior does not match any membership criteria of all the existing roles, the actor is added to the *default* role.

Upon observing the $send$ or $ready$ event from a computation actor belonging to a role group, the role propagates these events to coordinators to inquire about corresponding inter-role constraints. Unlike the original messages sent from actors, the message parameters in these events may contain extra information, such as the names of the sender and receiver actors, and their currently attached roles. This information helps the coordinator to determine what constraints need to be propagated to which role.

After observing the $propSend$ or $propReady$ event propagated from the roles, a coordinator checks its constraint store and locates the corresponding constraints, which may depend on both the message parameters and the coordinator's own states. The coordinator then enacts these constraints by sending a $tell$ event to the roles.

The formal operational semantics of the ARC model is given in [44].

4 Framework

In this section, we briefly describe several critical design issues of the framework, and then present the design in detail, along with a prototype implementation of the ARC model. Finally, we show the results of experiments demonstrating the scalability and performance overhead of the framework.

4.1 Design issues

The main design and implementation concern of the ARC framework is to provide the abstractions that implement the Actor, Role and Coordinator semantics, and at the same time provide good performance, scalability and flexibility for different applications. Based on this goal, there are several design issues we need to consider:

Implement coordination actors and events.

According to the definition of the ARC model, roles and coordinators are "coordination actors" communicating through event broadcasts. Therefore, we need to explicitly distinguish events and messages in the implementation.

As defined in [1, 2], computation actors are autonomous and active entities that communicate with each other through asynchronous messages, as are coordination actors. However, unlike computation messages that communicate among actors in a point-to-point fashion, events are broadcast to all coordination actors. Furthermore, events have a higher priority than computation messages. This ensures that messages that need to be coordinated will be manipulated by coordination actors before they are dispatched on computation actors. In both our model and implementation framework, the generation and consumption of events are treated as atomic behaviors and are enforced by using synchronization protocols.

Maintain scalability and performance as the number of entities increases.

One of the characteristics of ODE systems is that they usually have large numbers of computational entities. The introduction of active roles into the ARC model helps mitigate the scalability issues in coordination management by allowing coordinators to only coordinate roles, while roles only coordinator actors that share the same behaviors.

Because coordination in the ARC model is enforced transparently on the underlying actors, two problems may occur when the number of actors increases. First, every coordinated message triggers at least one event that must be handled by remote coordination actors. This may bring additional communication overhead. Second, roles and coordinators become potential bottlenecks, which may degrade performance and make systems hard to scale.

To alleviate these problems, we have developed a decentralized architecture to further distribute coordination behaviors and states to local physical nodes, thus avoiding bottlenecks and communication overhead. Because both roles and coordinators are active and stateful entities, multiple update and query operations may concurrently be applied to the states of those distributed replicas. Therefore, a synchronization protocol must be in place to ensure the consistency of the states among different nodes. If such synchronizations occur very frequently, the overhead of achieving synchronizations may exceed the benefit of distributing roles and coordinators to local platforms. Hence, tradeoffs need to be made to balance the communication and synchronization costs. Whether distributing the coordinator/role states will have performance gains is application dependent.

Avoid re-inventing the wheel to solve common problems.

Instead of developing our framework from scratch, we take advantage of existing technologies and tools to support distributed communication, i.e., distributed naming, synchronous and asynchronous communication, and locking schemes.

In the next section, we give the details of our framework's design, taking into account the above issues and providing our solutions to them.

4.2 An ARC framework

Figure 5 gives an overview use case diagram depicting the functional requirement from three categories of users in the system, i.e. the actors, roles, and coordinators. From this figure, the part within the dashed line box represent the functionality of traditional Actor system. By importing the concepts of role and coordinator, the use cases in ARC system become richer. The purpose of the framework is therefore to fulfill the functional requirements indicated in this use case diagram, while taking into account the design issues presented in Section 4.1.

The ARC framework is built on top of TAO (v1.4.1) [45], an implementation of the CORBA 3.x specification. To minimize the overhead and footprint of the ARC framework, we only use a small subset of services provided by TAO. Actors in the ARC framework are built as CORBA objects. They register themselves and locate other actors through the CORBA naming service, and communicate with each other through the TAO asynchronous message service. Figure 6 outlines the architecture of the framework. The Role Representative and Coordinator Representative objects localize the functionalities of coordination-actors to further increase scalability of the system. These concepts will be discussed in detail in a later in this section.

4.2.1 Actor platform and message manager

In the framework, an *Actor Platform* is installed on every physical node. It provides a uniform way to create actors and register actors as CORBA services. An Actor Platform is implemented as a "system actor" that creates actors, roles, and coordinators, initializes their states and behaviors, sends messages, and generates events.

With each actor creation, the Actor Platform also creates a Message Manager object for each actor (including both computation and coordination actors) to handle actor communication tasks. When an actor tries to send a message to another actor, it delegates the message to its Message Manager. For the sending actor, the Message Manager acts as a CORBA client object to send the message asynchronously to the destination actor's message manager, which acts as a CORBA server object. The receiving message manager then forwards the message to the receiving actor for processing. Thus, the CORBA middleware details are encapsulated in the implementation of the message manager and are transparent to application developers who use actors.

4.2.2 Modes

In our framework, users have the option to have logically remote coordinators and roles physically distributed to local Actor Platforms to reduce the communication overhead. Therefore, we provide three modes:

Fully Centralized Mode (FCM) In this mode, every coordination message has to go through potentially remote roles and remote coordinators. This mode is suitable for applications that require very frequent state updates in both coordinators and roles.

Partially Distributed Mode (PDM) The coordinator is distributed to the nodes where the coordinated roles are located, but roles are not distributed to the actor platforms. Therefore coordination requests from local nodes have to go through possibly remote roles, but these roles use local coordinator representatives instead of remote coordinators. Applications that do not anticipate frequent state updates in coordinators will benefit by using this mode.

Fully Distributed Mode (FDM) Both coordinator and roles are distributed to every related node. This mode brings best performance for applications with less frequent synchronization needs.

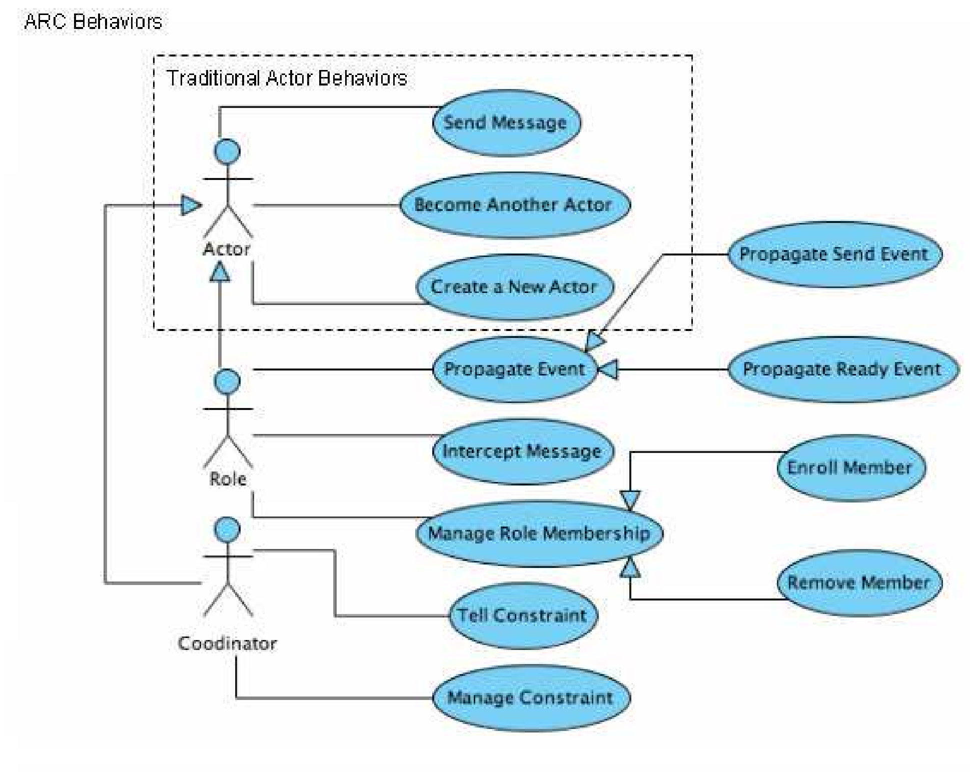


Figure 5: ARC Use Case Diagram

In the framework, we define two supporting entities: Coordinator Representative and Role Representative. As their names suggest, they represent coordinators and roles and perform coordination behaviors in local Actor Platforms. To facilitate deploying different modes, these representatives are implemented as coordination-actors. According to the definitions of coordination actors, they are able to communicate with each other through event communications. Based on the currently applied mode, different Coordinator Representative and Role Representative instances are bound to these interfaces during runtime and have different responsibilities. The relationship among Message Manager, Role, Coordinator, representative interfaces and their instances is depicted in Figure 7.

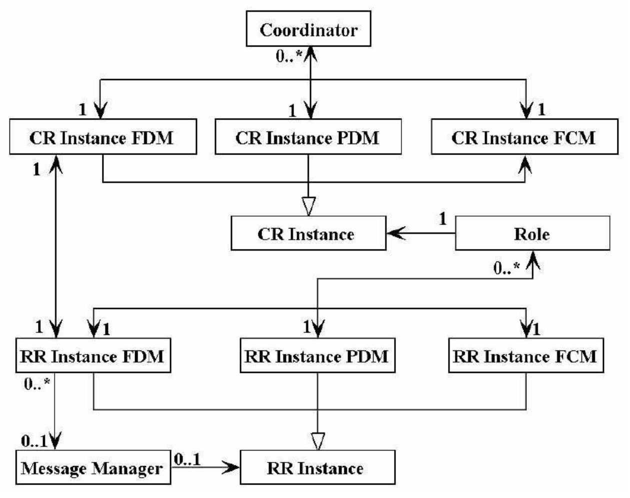


Figure 7: Multi-mode Class Diagram

4.2.3 State synchronization

In situations when synchronization is required among representatives, we apply the primary-backup and two-phase locking (2PL) protocol. The coordinators and roles are responsible for synchronizing the updates with their representatives distributed among other actor platforms. In the primary-backup protocol, these coordinators or roles act as primary objects and the representatives are backups. The Concurrency Service provided by TAO enables the primary objects to obtain and release locks in the 2PL algorithm.

4.2.4 Fully distributed mode implementation

In this paper we focus on the implementation of the Fully Distributed Mode (FDM). The implementations of the Partially Distributed Mode and Fully Centralized Mode are very similar and can be easily inferred from the current introduction.

With FDM, the local Actor Platform creates a Role Representative coordination actor for every existing role to ful-

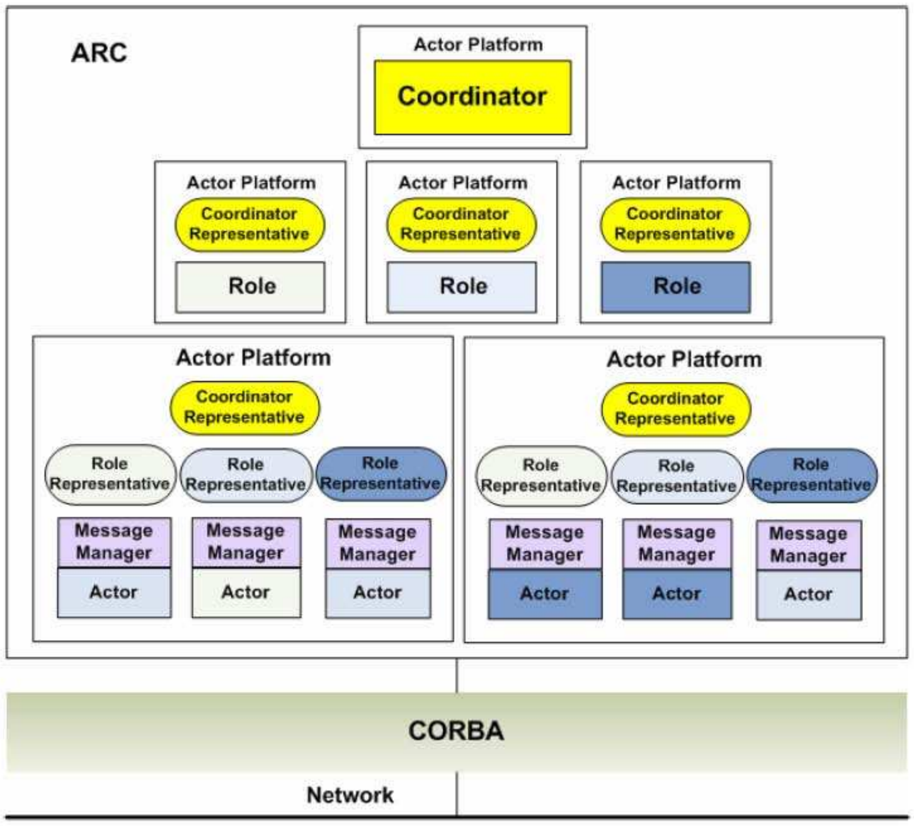


Figure 6: The Architecture of the ARC Framework

fill both its membership management behavior and coordination behavior. In the ARC model, it is the roles, but not the actors, that manage group membership. Whenever a new actor is created or an actor changes its behavior, the roles apply their *bind* and *unbind* operations to maintain the consistency of the membership. Figure 8 demonstrates the procedure of a Role Representative performing membership management and implementing the binding mechanism.

In the ARC model, coordination constraints are transparently applied to actors. This is achieved by (1) buffering the messages in receiver actors' mailboxes via Message Managers, (2) obtaining coordination constraints by forwarding events to the corresponding role representatives and coordinator representative for constraint checks, and (3) applying the coordination constraints by manipulating the messages in the mailboxes. The communication between two actors is shown in Figure 9.

If a constraint is found in its local store, the Role Representative requires the corresponding Message Manager to enact the constraint on the actor. As all these operations are performed locally and no remote communication is required, the constraint propagations do not introduce much performance overhead.

4.3 Evaluation

We have developed a prototype of the ARC framework. The experimental settings are as following: We have two Intel x86 machines. The first machine is a Pentium IV 1.7 GHz with 512MB RAM and the second is a Pentium IV 3.06GHz with 1GB RAM. Both of them are running Windows XP and connect with each other through a 100M ethernet switch. In our experiments, we developed a simple Ping-Pong application, that asks two actors, the Ping actor and the Pong actor, in different machines to continuously send and reply to a specific number of messages to each other.

Figure 10 shows the performance comparisons between the Actor Architecture (AA) framework [22] and the ARC framework. AA is an actor-based framework developed by Agha's group at UIUC. AA is implemented in Java and provides its own ad-hoc solutions to core distributed applica-

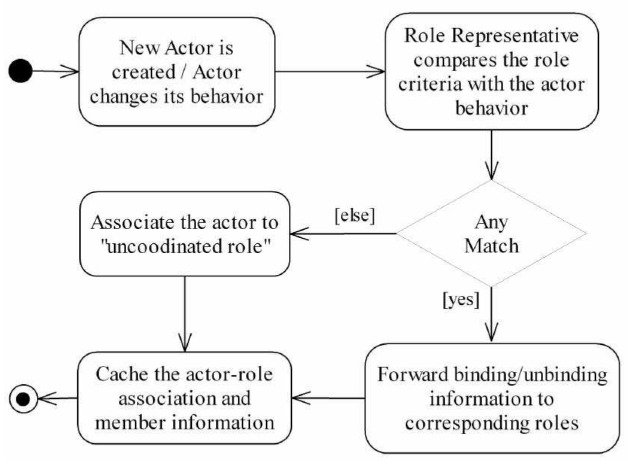

New Actor is created / Actor changes its behavior
Role Representative compares the role criteria with the actor behavior
Any Match
[else]
[yes]
Associate the actor to "uncoodinated role"
Cache the actor-role association and member information
Forward binding/unbinding information to corresponding roles

Figure 8: Actor-Role Binding Mechanism

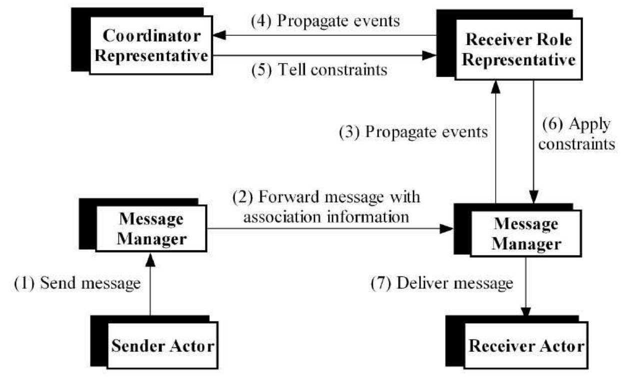


Figure 9: Communication between Coordinated Actors in FDM

tion features, such as the Naming Service. The ARC framework is implemented in C++ and utilizes CORBA services.

In this test we use AA and ARC to send messages (with a size of 100 bytes) between two actors on different machines. From figure 13, we can see that ARC outperforms AA in actor communications. The average throughput by using ARC is 85% higher than using AA. This is mainly because the Java Virtual Machine brings heavy overhead to AA. In addition, the optimized naming and communication services provided by TAO also improve the ARC framework's communication performance.

Figure 11 demonstrates the situation when multiple actor pairs run and send messages concurrently. We ran up to 100 actors on each machine. These pairs of actors sent messages and replied to them concurrently. As the figure shows, increasing the number of actors had little impact on the performance of the ARC. However, the figure also shows an 'unintuitive' result: the performance of the 40 actor case is better than the 20 and 10 actor cases. This happened because the larger number of actors increases the odds that messages will share transportation connections.

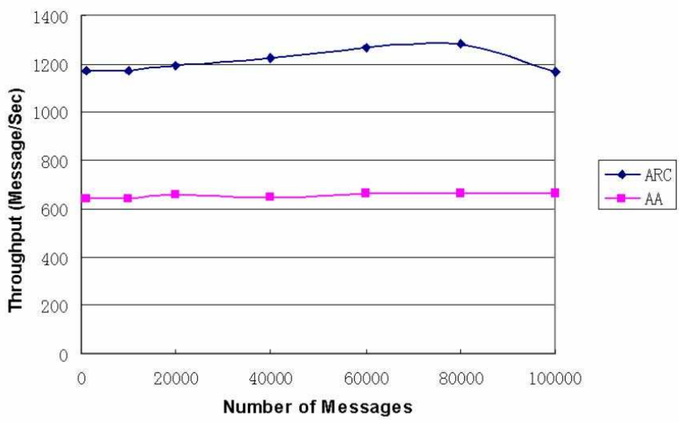


Figure 10: ARC vs. AA on actor communication

This reduced the overhead of opening and closing connections. Once the number of actors is greater than 40, the connections are saturated and the performance becomes stable.

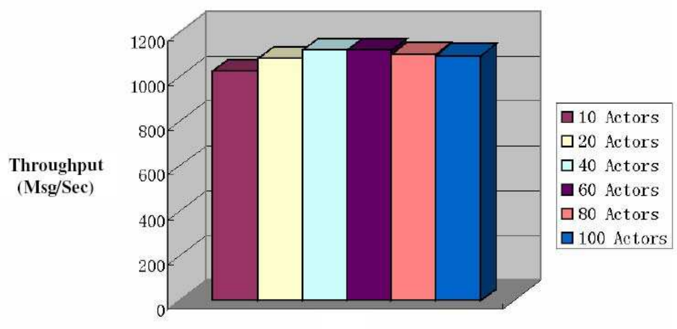


Figure 11: Performance of ARC when the number of actors increases

In the previous experiments, we performed actor communications without considering coordination constraints. When coordination requirements are taken into account, actors need to collaborate with each other to achieve system requirements. For example, in the Ping-Pong application, we could have an extra mutual exclusion requirement that at any time only one pair of Ping and Pong actors can send messages to each other. To satisfy this requirement, the Ping and Pong actors that want to send messages have to communicate with each other to make sure that there are no other actors competing for the permission to send a message. If there is more than one actor seeking permission, a decision needs to be made about which one gets permission first. This will require communications to be sent, and some kind of election protocol to be followed.

If there are n Ping actors competing for permission to send a message, then there will be at least $2n(n-1)$ [48] communication messages to achieve synchronization before a message can be sent out. This is a typical synchronization problem for networking and distributed environments. An obvious solution is to use an explicit coordinator to synchronize the "sending message" requirements among actors. To achieve the same synchronization with an explicit coordinator requires most $3n$ [48] extra communication messages. Thus, in an ODE system or similar environment where the number of actors is large and coordination among them is frequent, an explicit coordinator can drastically reduce communication overhead and improve scalability. Figure 12 depicts the difference between the solution using a coordinator, which is represented by a star topology, and the one without an explicit coordinator, which is represented by a mesh topology.

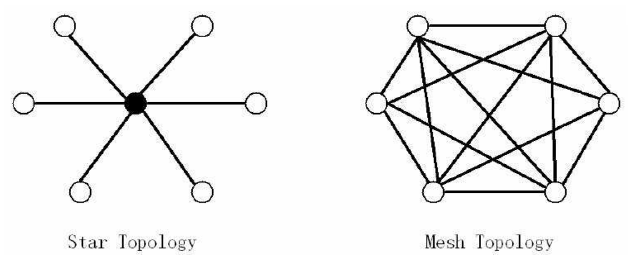


Figure 12: With or Without Coordinator - a Topology View

From the above analysis, it is clear that adding an extra coordinator layer actually increases performance when the number of actors is large and coordination among them is unavoidable. The next question to ask is if adding an extra layer (the role layer) will seriously degrade the performance of the system with a single coordinator layer. We test this by introducing role coordination entities to achieve *intra-role* coordination constraints. The current test case is under FDM and follows the procedure demonstrated in Figure 9.

In this experiment, we arranged for $10,000$ messages to be sent between two actors on different machines. We also provided both *inter-role* and *intra-role* constraints. After introducing two roles, the PingRole and the PongRole, we divided the constraints into three categories: 20% became inter-role constraints stored in a coordinator, 40% became *intra-role* constraints stored in PingRole, and the remaining 40% were *intra-role* constraints stored in PongRole. Constraint checks were simulated using simple string comparisons. Figure 13 gives the measurements.

As shown in Figure 13, when there are 100 constraints in a single coordinator, the overhead of introducing two extra role entities is about 3.5%; when there are 500 constraints, the overhead is about 2.7%. The main overhead comes from the two extra communications between the sender and receiver actors and their attached roles. This number is fixed no matter how many constraints need to be checked. The total number of constraints is the same in two situations. When there are no roles, a coordinator has to check all these constraints; in contrast, when there are two extra roles, the coordinator only handles 20% of the constraints, and rest of the constraints are handled by the two roles concurrently. As a result, the overhead actually

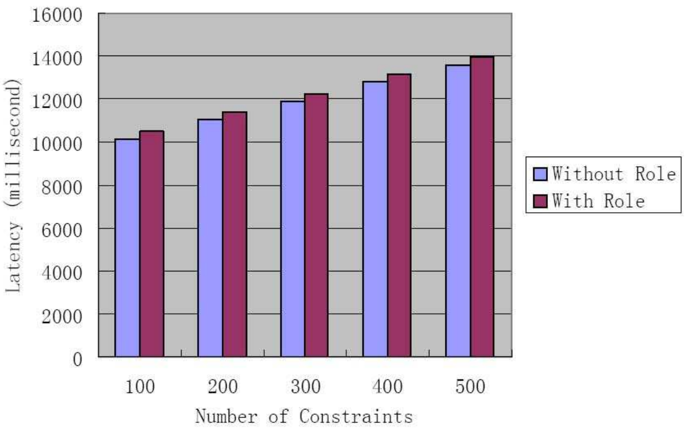


Figure 13: Overhead of introducing role layer

decreases when the number of constraints increases.

Though these tests have not conducted on FCM and PDM, we can expect by looking at their descriptions that because roles and actors live in different nodes, the communication overhead will be larger than in the current FDM test. However, in such modes, synchronization overhead will become the focus of the application and trade-offs are made to satisfy that.

Finally, we look at the modularity brought by separating the coordination layers from the underlying computational logic in the ARC framework. We demonstrate this by introducing an extra requirement for the Ping-Pong application: after a Pong actor receives a message in its mailbox from the Ping actor, it has to wait for a specific period of time $t1$ before it sends back a response. The following gives the pseudo-code that enforces this timing constraint in the Pong actor:

```
HandleMsg(String message, Int waitTime){
  Message msg = parse(message);
  String responseMsg = getResponse();
  if (msg.SenderTypeName == "Ping") {
     wait(waitTime);}
  send(Ping, responseMsg);
}
```

If we have multiple Pong actors in the system, then we will have to add this code to every Pong actor to maintain the timing constraint. Furthermore, if in the future we want to modify the time period, for example from $t1$ to $t2$, we have to update the codes for all the Pong actors, change the constraints and re-compile them. But if we have an explicit coordination actor, we can use it to specify these timing constraints.

Computational logic is separated from coordination constraints, and can be developed independently. Below shows the code with the ARC model. The `HandleMsg` is the code in a Pong actor to implement the "response" logic, and the `HandleEvt` is the code in a coordinator to specify the timing constraint. The timing constraint is further enforced by a role that reroutes a message in the Pong actor's mailbox to a sink for a period of time `waitTime` before dispatching it for processing. This coordination operation is transparent to the actor computation, and we can modify such constraints without affecting the underlying computational logic.

```
HandleMsg(String message){
  Message msg = parse(message);
  String responseMsg = getResponse();
  send(Ping, responseMsg);
}

HandleEvt(String event, Int waitTime){
  Event evt = parse(event);
  if (evt.eventTypeName == "PropSendMsg")
   if (evt.senderRoleTypeName ==
                              "PingRole")
      tell(PongRole,
          "reroute, sink, waitTime");
}
```

5 Conclusion

In this paper, we presented a framework based on the ARC model to support the development of ODE applications. The ARC model is a role-based and decentralized coordination model. Under this model, a system's QoS requirements are treated as coordination concerns and are separated from concurrent computation logic. The coordination constraints are imposed on computations through message manipulations that are transparent to the computation itself. In addition, to address the dynamicity and the openness inherent in an ODE system, we introduced active roles that not only provide abstractions for actor functional behaviors, but also take part in the coordination activities. Hence, the coordination subsystem itself becomes distributed and thus inherits all the benefits a distributed system may offer.

The framework provides an interface to allow users to create actors, roles and coordinators. Based on detailed application requirements, the framework distributes the coordinators and roles and collocates them with local actors so that both performance and scalability can be improved. In addition, the framework also provides efficient mechanisms to support automatic and runtime role group management, and message management. Our prototyping and empirical experiments have shown that we are able to achieve role-based coordination with limited performance overhead. The experiments also indicate that the framework scales well when the number of entities involved in the system increases.

Our future work is to apply the ARC model and its realization to help mitigate the difficulties in developing practical QoS aware applications in ODE systems. Such systems may have multiple dimensions of QoS requirements such as real-time, fault tolerance, energy consumption, and security constraints, etc. To be more specific, we want to extend our framework to combine resource management, real-time features and fault tolerance mechanisms, so that multiple non-orthogonal QoS requirements can be studied and supported based on a uniform coordination model. To achieve this, we plan to use classic ODE applications, such as a simulation of a simplified Air Traffic Control (ATC) system, as cases studies to demonstrate and evaluate the advantages of the model and the framework.

Acknowledgement

This work is supported by NSF under grant CNS 0746643.

References

[1] G. Agha. *Actors: A model of concurrent computation in distributed systems*. MIT Press, 1986.

[2] G. Agha, I. A. Mason, S. F. Smith, and C. L. Talcott. A foundation for actor computation. *Journal of Functional Programming*, 7(1):1–72, 1997.

[3] M. Aksit, B. Tekinerdogan, and L. Bergmans. The six concerns for separation of concerns. In *Workshop on Advanced Separation of concerns*, 2001.

[4] F. Arbab. IWIM: A communication model for cooperative systems. In *The 2nd International Conference on the Design of Cooperative Systems*, pages 567–585, 1996.

[5] F. Arbab. A foundation model for components and their composition. Technical report, CWI, Amsterdam, Netherlands, 2004.

[6] F. Arbab. Reo: a channel-based coordination model for component composition. *Mathematical Structures in Computer Science*, 14(3):329–366, 2004.

[7] Barbaran, Diaz, Esteve, Garrido, Llopis, and Rubio. A real-time component-oriented middleware for wireless sensor and actor networks. *cisis*, 00:3–10, 2007.

[8] A. Berry and S. Kaplan. Open, distributed coordination with finesse. In *The 1998 ACM Symposium on Applied Computing*, pages 178–184, 1998.

[9] G. Cabri, L. Ferrari, and L. Leonardi. Brain: a framework for flexible role-based interactions in multiagent systems. 2888:145–161, 2003.

[10] N. Carriero and D. Gelernter. Linda in context. *Communications of the ACM*, 32(4):444ÍC458, 1989.

[11] A. Colman and J. Han. Coordination systems in role-based software. 3454:63–78, 2005.

[12] Riccardo Crepaldi, Albert Harris III, Rob Kooper, Robin Kravets, Gaia Maselli, Chiara Petrioli, and Michele Zorzi. Managing heterogeneous sensors and actuators in ubiquitous computing environments. In *First ACM Workshop on Sensor Actor Networks*, 2007.

[13] J. C. Cruz. Opencolas: A coordination framework for colas dialects. 2315:231–247, 2002.

[14] D.F. Ferraiolo and D.R. Kuhn. Role based access control. In *The 15th National Computer Security Conference*, 1992.

[15] M. Fowler. Dealing with roles. In *European Conference on Pattern Language of Programs*, 1997.

[16] S. Frølund. *Coordinating Distributed Objects: An Actor-Based Approach to Synchronization.* MIT Press, 1996.

[17] E. Gamma, R. Helm, R. Johnson, and J.Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional Computing Series, 1995.

[18] W. Harsch and C. V. Lopes. Separation of concerns. Technical report, Northeastern University technical report NU-CCS-95-03, Boston, 1995.

[19] Carl Hewitt. large-scale organizational computing requires unstratified paraconsistency and reflection. In *International Conference on Autonomous Agents and Multiagent Systems*, 2007.

[20] Carl Hewitt. What is commitment? physical, organizational, and social. *LNAI*, 4386, 2007.

[21] C.A.R. Hoare. *Communicating Sequential Processes*. Computer Science. Prentice Hall International, 1985.

[22] M. W. JANG. The actor architecture manual, 2004.

[23] Woochul Kang and Sang H. Son. The design of an open data service architecture for cyber-physical systems. *AC SIGBED Review*, 5(1), 2008.

[24] Rajesh K Karmani and Gul Agha. Debugging wireless sensor networks using mobile actors. In *RTAS Poster Session*, 2008.

[25] E. A. Kendall. Role modeling for agent system analysis, design and implementation. *IEEE Concurrency*, 8(2):34–41, 2000.

[26] Ozcan Koc, Chaiporn Jaikaeo, and Chien-Chung Shen. Navigating actors in mobile sensor actor networks. In *First ACM Workshop on Sensor Actor Networks*, 2007.

[27] B. Kristensen and K. Osterbye. Roles: conceptual abstraction theory and practical language issues. *Theory and Practice of Object System*, 3(2):143–160, 1996.

[28] YoungMin Kwon, Sameer Sundresh, Kirill Mechitov, and Gul Agha. Actornet: An actor platform for wireless sensor networks. In *Fifth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS), pages 1297-1300*, 2006.

[29] Youngmin Kwon, Sameer Sundresh, Kirill Mechitov, and Gul Agha. Actornet: An actor platform for wireless sensor networks. In *In Proc. of the 5th Intl. Joint Conf. on Autonomous Agents and Multiagent Systems (AAMAS).* AAMAS, 2006.

[30] E. A. LEE. What's ahead for embedded software. *IEEE Computer*, 33(9):18–26, 2000.

[31] J. LIU, J. EKER, J. W. JANNECK, X. LIU, and E. A. LEE. Actor-oriented control system design: A responsible framework perspective. *IEEE Transactions on Control System Technology*, 12(2):250–262, 2004.

[32] José Meseguer and Carolyn Talcot. Semantic models for distributed object reflection. In *European Conference on Object-Oriented Programming, ECOOP'2002, LNCS 2374*, pages 1–36, 2002.

[33] R. Milner. *Communicating and Mobile Systems: the Pi-Calculus.* Cambridge University Press, 1999.

[34] N. H. Minsky and V. Ungureanu. Law-governed interaction: a coordination and control mechanism for heterogeneous distributed systems. *ACM Transactions on Software Engineering Methodology*, 9(3):273–305, 2000.

[35] Jayadev Misra and William R. Cook. Computation orchestration: A basis for wide-area computing. *Journal of Software and Systems Modeling*, May 2006.

[36] Rocco De Nicola, Gianluigi Ferrari, and Rosario Pugliese. Klaim: a kernel language for agents interaction and mobility. *IEEE Transactions on Software Engineering*, 24:315–330, 1998.

[37] R. Nicola de, J.P. Katoen, D. Latella, and M. Massink. Towards a logic for performance and mobility. 2005.

[38] A. Omicini and E. Denti. Formal respect. *Electronic Notes in Theoretical Computer Science*, 48:179–196, 2001.

[39] A. Omicini, A. Ricci, and M. Viroli. Agent coordination contexts for the formal specification and enactment of coordination and security policies. *Science of Computer Programming*, 63(1):88–107, 2006.

[40] A. Omicini and F. Zambonelli. Tuple centres for the coordination of internet agents. In *The ACM Symposium on Applied Computing*, pages 183–190, 1999.

[41] G. A. Papadopoulos and F. Arbab. Coordination models and languages. *Advances in Computers*, 46:330–401, 1998.

[42] Gian Pietro Picco, Amy L. Murphy, and Gruia catalin Roman. Lime: Linda meets mobility. In *21st Internation Conference on Software Engineering*, pages 368–377. ACM Press, 1999.

[43] José Proença and Dave Clarke. Coordination models orc and reo compared. *Electronic Notes in Theoretical Computer Science*, 194(4):57–76, 2008.

[44] S. Ren, N. Chen, Y. Yu, P.-E. Poirot, L. Shen, and K. Marth. Actors, roles and coordinators a coordination model for open distributed embedded systems. 4038:247–265, 2006.

[45] D. C. Schmidt. The design of the tao real-time object request broker. In *Computer Communications*, 1998.

[46] Carolyn Talcott, Marjan Sirjani, and Shangping Ren. Ccoordinating asynchronous and open distributed systems under semiring-based timing constraints. *Electronic Notes in Theoretical Computer Science*, 2008.

[47] Carolyn L. Talcott. Coordination models based on a formal model of distributed object reflection. *Electronic Notes in Theoretical Computer Science*, 150:143–157, 2006.

[48] A. S. Tanenbaum and M. V. Steen. *Distributed Systems - Principles and Paradigms*. Prentice Hall. Upper Saddle River, New Jersey, 2002.

[49] C. A. Varela and G. A. Agha. Towards a discipline of real-time programming. *Communications of the ACM*, 20(8):577–583, 1977.

[50] N. Venkatasubramanian, G. A. Agha, and C. Talcott. A metaobject framework for qos-based distributed resource management. In *The Third International Symposium on Computing in Object-Oriented Parallel Environments*, 1999.

[51] Yue Yu, Shangping Ren, and Carolyn Talcott. Comparing three coordination models: Reo, arc. *Electronic Notes in Theoretical Computer Science*, (16956), 2008.

Analysis of an Immune Algorithm for Protein Structure Prediction

Andrew J. Bennett, Roy L. Johnston, and Eleanor Turpin
University of Birmingham, Birmingham, United Kingdom
E-mail: ajb@tc.bham.ac.uk

Jun Q. He
Aberystwyth University, Aberystwyth, United Kingdom

Keywords: HP lattice bead model, immune algorithm, population diversity tracking, protein modelling

Received: June 23, 2008

The aim of a protein folding simulation is to determine the native state of a protein from its amino acid sequence. In this paper we describe the development and application of an Immune Algorithm (IA) to find the lowest energy conformations for the 2D (square) HP lattice bead protein model. Here we introduce a modified chain growth constructor to produce the initial population, where intermediate infeasible structures are recorded, thereby reducing the risk of attempting to perform wasteful point mutations during the mutation phase. We also investigate various approaches for population diversity tracking, ultimately allowing a greater understanding of the progress of the optimization.

Povzetek: V članku je opisan razvoj in izvedba imunskega algoritma (IA) za iskanje najnižje energijske strukture za 2D (kvadratne) HP mrežno nanizanega modela proteina.

1 Introduction

Predicting the 3-dimensional secondary and tertiary structure of a protein molecule from its (primary structure) amino acid sequence alone is an important problem in chemical biology [1]. Under certain physiological conditions, the amino acid chain will reliably fold into a specific native state (biologically active conformation). The protein folding problem is the search for this native state for a given sequence of amino acid residues. The reliability of protein folding is said to be dominated by the presence of a "folding funnel" on the folding energy landscape since systematic or random searching is clearly infeasible for large numbers of amino acids [2]. Therefore, discovering the nature of the folding energy landscape is necessary to develop a better understanding of the folding dynamics [3].

Many protein models have been developed, ranging from simple, minimalist models such as the HP lattice bead model [4], to more complicated and computationally expensive models such as off-lattice interpretations. The most common lattice structures are 2D square and 3D cubic. More computationally intense models include the dynamical lattice and all-atom models, both introducing more complicated fitness functions.

In this work, the standard HP lattice bead model has been incorporated into an immune algorithm. Despite the minimalistic approach employed by this model, it has been shown to belong to the "NP-Hard" set of problems [5]. Monte Carlo [6], chain growth algorithms [4], simulated annealing [7], genetic algorithms [5, 8, 9], ant colony optimization [10] and more recently immune algorithms [11] have been developed by many researchers as heuristic and approximate solutions for this and other computationally hard problems.

2 Methodology

2.1 The HP lattice bead model

In this work, the standard HP lattice bead model is embedded in a 2-dimensional square lattice, restricting bond angles to only a few discrete values [4]. Interactions are only counted between topological neighbours, that is between beads (representing amino acids) that lie adjacent to each other on the lattice, but which are not sequence neighbours [3]. The energies corresponding to the possible topological interactions are as follows:

$$\epsilon_{HH} = -1.0 \qquad \epsilon_{HP} = 0.0 \qquad \epsilon_{PP} = 0.0 \tag{1}$$

By summing over these local interactions, the energy of the model protein can be obtained:

$$E = \sum_{i<j} \epsilon_{ij} \Delta_{ij} \tag{2}$$

where

$$\Delta_{ij} = \begin{cases} 1 & \text{if i and j are topological neighbours,} \\ & \text{but are not sequence neighbours;} \\ 0 & \text{otherwise.} \end{cases}$$

The HP lattice model recognises only the hydrophobic interaction as the driving force in protein folding, with

Name	Length	E^*	Sequence
HP-18a	18	-9	$PHP_2HPH_3PH_2PH_5$
HP-18b	18	-8	$HPHPH_3P_3H_4P_2H_2$
HP-18c	18	-4	$H_2P_5H_2P_3HP_3HP$
HP-20a	20	-10	$H_3P_2(HP)_2HP_2(HP)_2HP_2H$
HP-20b	20	-9	$HPHP_2H_2PHP_2HPH_2P_2HPH$
HP-24	24	-9	$H_2P_2(HP_2)_6H_2$
HP-25	25	-8	$P_2HP_2(H_2P_4)_3H_2$
HP-36	36	-14	$P_3H_2P_2H_2P_5H_7P_2H_2P_4H_2P_2HP_2$
HP-48	48	-23	$P_2H(P_2H_2)_2P_5H_{10}P_6(H_2P_2)_2HP_2H_5$
HP-50	50	-21	$H_2(PH)_3PH_4P(HP_3)_3P(HP_3)_2HPH_4(PH)_4H$

Table 1: Benchmark HP sequences used in the present study [12]. The lowest energies that have been found for these sequences are indicated by E^*.

many native structures protecting the hydrophobic core with polar residues, resulting in a compact arrangement [11]. This idea reflects the repulsive nature of the interactions between the hydrophobic residues and the surrounding water molecules [3].

2.2 The coordinate system

A previous study has illustrated how a local coordinate system offers better performance than a global one for studying protein folding [2]. In this work, a *local coordinate system* is used to define the folding conformation of the model proteins, that is the position of bead j is defined relative to beads $(j-1)$ and $(j-2)$. As the energy is identical for rotationally related structures, the bond between the first two beads lies along the x-axis, with these beads having coordinates (0,0) and (1,0) respectively. As a result, the search spaced is halved. The bond joining the $(j-1)^{th}$ and j^{th} beads can be left, right or straight ahead relative to the bond joining the $(j-2)^{th}$ and $(j-1)^{th}$ bead, corresponding to an integer representation of 0, 1 and 2 respectively. The protein conformation is therefore expressed as a *conformation vector*, containing a list of 0's, 1's and 2's.

For this study, a set of well investigated protein benchmark sequences have been considered: the tortilla HP benchmark sequences [12]. They range in length from eighteen to fifty beads and are listed in Table 1. The table also includes the energy, E^*, of the putative global minimum (or conformations, since all of these structures have degenerate global minima) for each sequence.

3 The immune algorithm

An immune algorithm [13] is inspired by the clonal selection principle employed by the human immune system. In this process, when an antigen enters the body, B and T lymphocytes are able to clone upon recognition and bind to it [13]. Many clones are produced in response and undergo many rounds of somatic hypermutation. The higher the fitness of a B cell to the available antigens, the greater the chance of cloning. Cells have a certain life expectancy, allowing a higher specific responsiveness for future antigenic attack [11].

The IA presented here includes the aging, cloning and selection operators used in a previous study by Cutello *et al.* [11], with modified constructor and mutation operators. The constructor employs a backtracking algorithm that records some of the possible mutations by testing bead placement during chain growth. These possibilities are exploited and updated during the mutation process, preventing an infeasible conformation from occurring based on the preceding self-avoiding structure for a particular point in the model protein chain. In retaining this information, infeasible mutations are not explored, allowing a greater number of constructive mutations to be investigated. Figure 1 illustrates the stages involved in placing two consecutive beads during the chain growth phase. Before committing a bead to the lattice, all possible directions are explored, 1(a), and from the valid options available, a random choice is made, 1(b). Again all possible choices are investigated, marking any infeasible options (note that choosing left will not result in a self avoiding conformation), 1(c), and a valid choice is selected from the remaining options, 1(d). Any remaining valid choices are left unmarked for use in the first mutation phase after the initial chain growth. Once a valid mutation has been made, the entire structure is reconstructed as before marking any infeasible directions as a result of the new conformation vector.

In the basic IA set up, there are a maximum of 10,000,000 fitness evaluations, with the maximum number of generations set to 500,000. In order to estimate the optimal combination of parameters, we adopted the procedure used by Cutello *et al.*, whereby the maximum B-Cell age and the number of clones were each varied from 1 to 10. Population sizes examined were 10, 25, 50, 100 and 200. This provided a combination of 500 different parameter sets for each sequence, which was applied to all the benchmark sequences up to 25 beads in length. All fitness evaluations for the best success rates were collated and graded for overall performance. As a result of this preliminary testing, the results presented below were obtained using a maximum B-Cell age of 4, 3 clones and a popu-

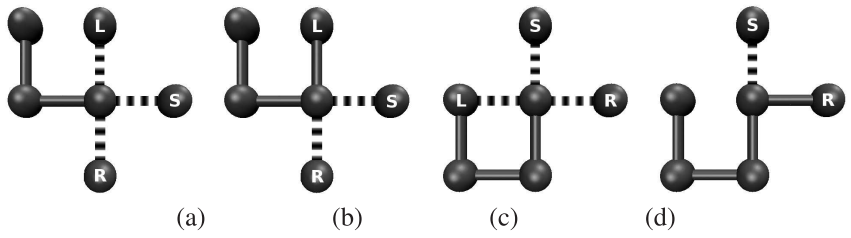


Figure 1: Stages of the chain growth algorithm investigating left (L), right (R) and straight (S) availability (a), random selection from all available directions (b), at the next locus investigation of L, R and S availability (c) and random selection from remaining available S and R directions (d).

lation size of 10. All results quoted are averaged over 30 independent runs.

4 Results

4.1 Algorithm comparison

With CPU time being hardware dependent, the number of fitness evaluations (together with the percentage success rate) have been used to assess the efficiency of the algorithm, as shown in Table 2 for the benchmark sequences.

It is apparent from Table 2 that, although the use of memory B-Cells [11] hinders the discovery of global minima for some of the smaller sequences, it enhances the search for the larger, more difficult to find sequences. The memory ability allows mid to high fitness conformations to remain in the population for a longer number of generations. For larger sequences, this allows a more detailed exploration in certain areas of the potential energy surface, permitting the memory B-Cells to converge towards the global solution much sooner. In contrast, for smaller sequences the mid to high fitness range is much smaller, thereby preventing a rapid exploration of the potential energy surface by retaining unfavourable segments of local structure for a larger number of generations. Generally, the use of memory B-Cells allows a more diverse inspection of the potential energy surface, due to a greater number of the degenerate conformations being found. This is achieved as favourable fragments of local structure are not rapidly disposed of during the retirement process, hindering efficiency as a consequence.

The algorithm presented here shows promising results, being comparable to the work of Cutello *et al.* [11]. While our success rates for the larger sequences (e.g. HP-48) are a little lower, in some cases our number of fitness evaluations show an improvement.

4.2 Analysis of global minima

The compact structural arrangement present in all global minima (GMs) is apparent from the example GMs shown in Fig. 2. With the driving force being the hydrophobic topological contact, it can be seen that compact hydrophobic cores give rise to high fitness conformations. Inspection of the HP-48 global minimum (i) allows us to understand the poor success rate for this sequence. The 5×5 hydrophobic core presents a problem to the IA (or other optimization algorithms [3]) in achieving convergence, as a single misplaced hydrophobic bead will result in only a metastable conformation. The problem does not exist for the HP-50 sequence (j), due to the presence of two small hydrophobic cores coupled by a chain of hydrophobic beads, which explains the higher success rate and fewer average structure evaluations necessary for HP-50, compared with HP-48 and (when using memory B-cells) even the much shorter HP-36 sequence [3]. The work of Cutello *et al.* supports this idea [11], as similar magnitudes of the number of fitness evaluations for these problematic sequences can be seen, with a much lower success rate for HP-48 than for any other instance.

4.3 Tracking population diversity

The much larger populations required to ensure population diversity can be problematic for both GAs and IAs. In this section, a single run, with population size 200 for sequence HP-20a has been analyzed. The global minimum was found in generation 28, at which point the algorithm was terminated due to meeting the search criteria. In order to help us understand the progress of the optimization and ultimately to improve the methodology, monitoring population diversity and the progress of the algorithm is beneficial.

Figure 3(a) assigns a colour to each of the three possible direction decisions (corresponding to alleles in a genetic sense) made when placing each successive bead. It can be seen that initial structure generation, using the IA's constructor, is indeed statistically uniform, showing the frequency of left (grey), right (light blue) and straight ahead (dark blue) choices at each locus of the model protein chain to be very similar. In contrast, Fig. 3(b) illustrates how this statistical distribution is skewed in the final population (generation 28), in that the IA has concentrated its search to a much narrower region of the potential energy surface. It should also be noted that position 6 in the chain has a very low frequency of the straight ahead choice (dark blue), because (for most population members) previous direction decisions preclude (for structural and/or energetic reasons) this choice from being made at this chain position.

Sequence	No Memory B-Cells		Memory B-Cells	
	%Success	No. Evaluations	%Success	No. Evaluations
HP-18a	100	89,578	100	117,251
HP-18b	100	40,167	100	200,740
HP-18c	100	87,761	100	72,270
HP-20a	100	26,207	100	312,405
HP-20b	100	15,221	100	30,414
HP-24	100	26,580	100	49,616
HP-25	100	79,042	100	95,123
HP-36	63	4,867,993	90	3,082,014
HP-48	3	6,318,721	3	4,195,086
HP-50	50	4,904,031	96	853,706

Table 2: Comparison of the percentage success and average number of structure evaluations with and without using memory B-Cells

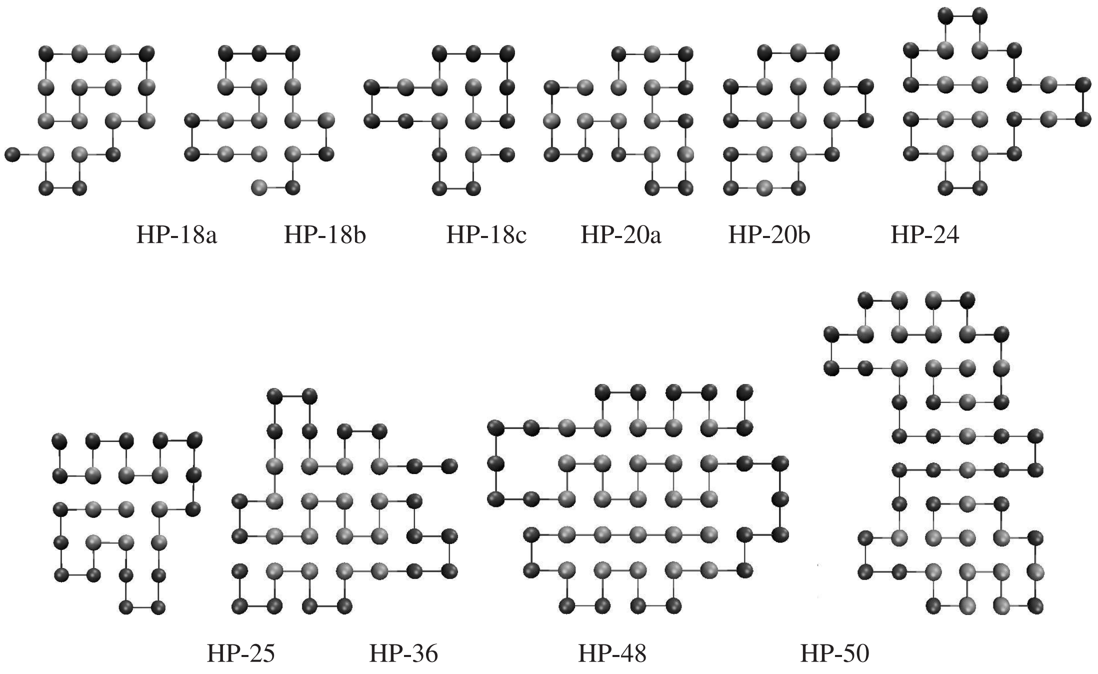
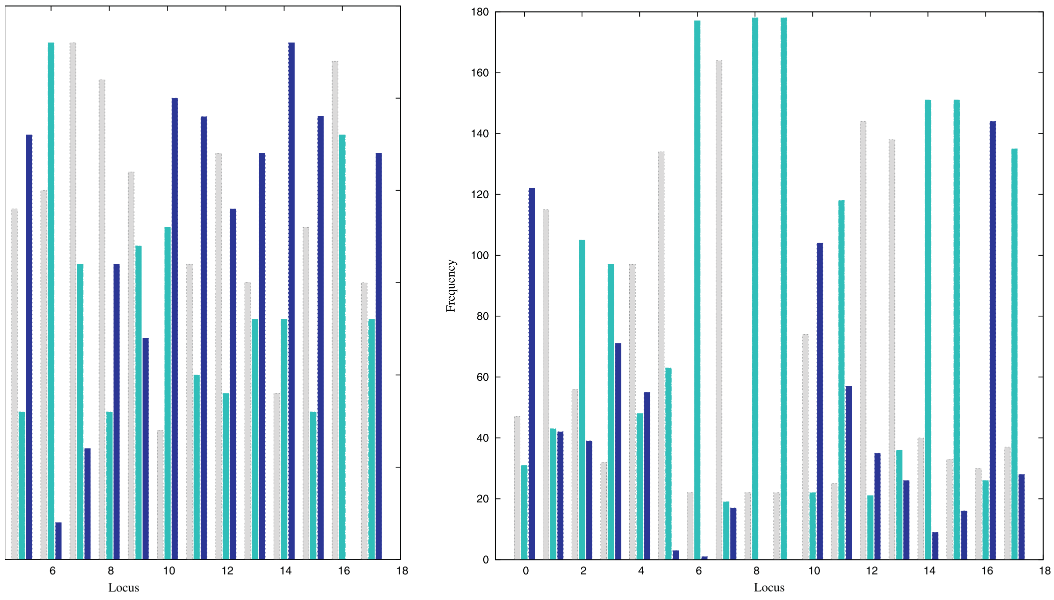


Figure 2: Examples of GM structures for the benchmark sequences.

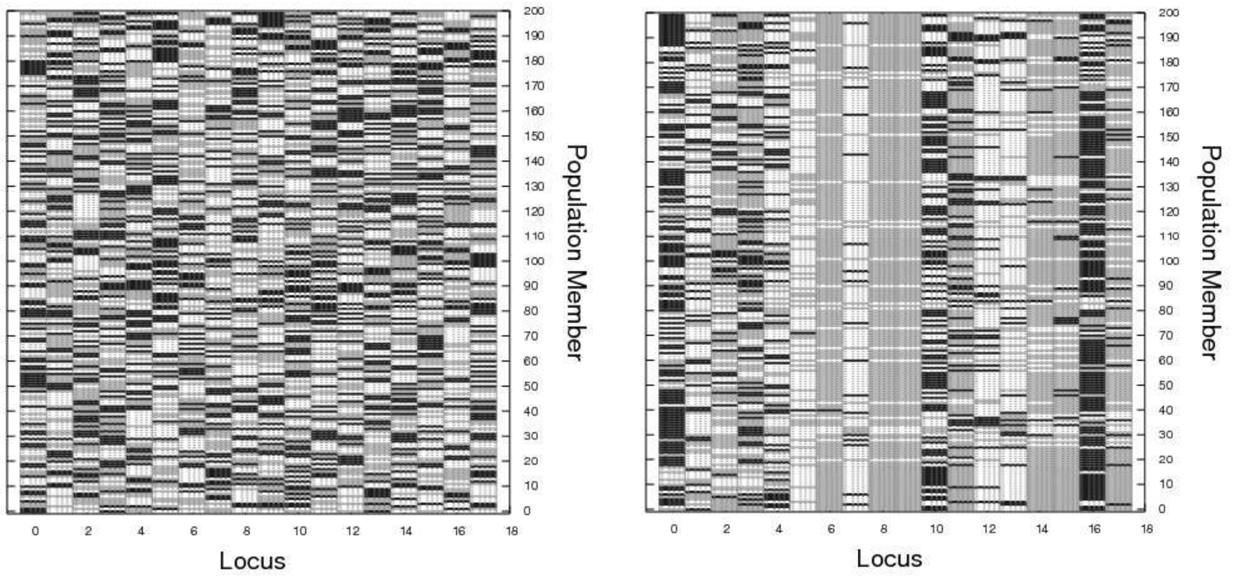
Figure 4 shows a graphical representation of the initial and final populations of the calculation. By plotting the conformation vector for each population member, the population can be quickly compared for diversity. Population members are ordered by descending fitness and the colour scheme is similar to the allele frequency distribution shown in Fig. 3, but with white replacing grey for the left choice. It is clear that initially the population has high diversity (in agreement with the allele frequency plot shown above), with the algorithm preserving favourable regions of local structure (corresponding to schemata in a GA sense) as the calculation converges. More detailed analysis of the final population shows that there are often correlations (or anti-correlations) between directions at specific loci, with certain combinations giving rise to favourable energies or infeasible structures, respectively.

For simple protein models such as the HP lattice bead model, the Hamming distance (d_H, which is the number of bit differences between two conformation vectors) can be used as a simple measure of similarity between structures in the population. Figure 5(a) plots the frequency of the Hamming distances between all pairs of structures in the population as a function of generation. (As the population size is 200, there are a total of 19,900 pair Hamming distances). It can be seen how the diversity of the population changes as the calculation approaches the global minimum (which is found in generation 28). Combining this with a plot of the best, worst and average fitnesses in the population, as a function of generation (Fig. 5(b)), it should be noted that structural diversity shows a more uniform spread (beginning around generation 20) as favourable segments of local structure begin to dominate the population, with the search focussing on a much more concentrated area of the potential energy surface. It is also evident that the



(a)　(b)

Figure 3: The frequency of alleles at each locus along the model protein chain for the initial population (a) and the final population (b), left (light grey), right (dark grey) and straight ahead (black).



(a)　(b)

Figure 4: Graphical representation of an initial population (a) and final population (b) of B-Cells, left (light grey), right (dark grey) and straight ahead (black). Population members are sorted by descending fitness, with structures of the highest energy at the bottom of the plot.

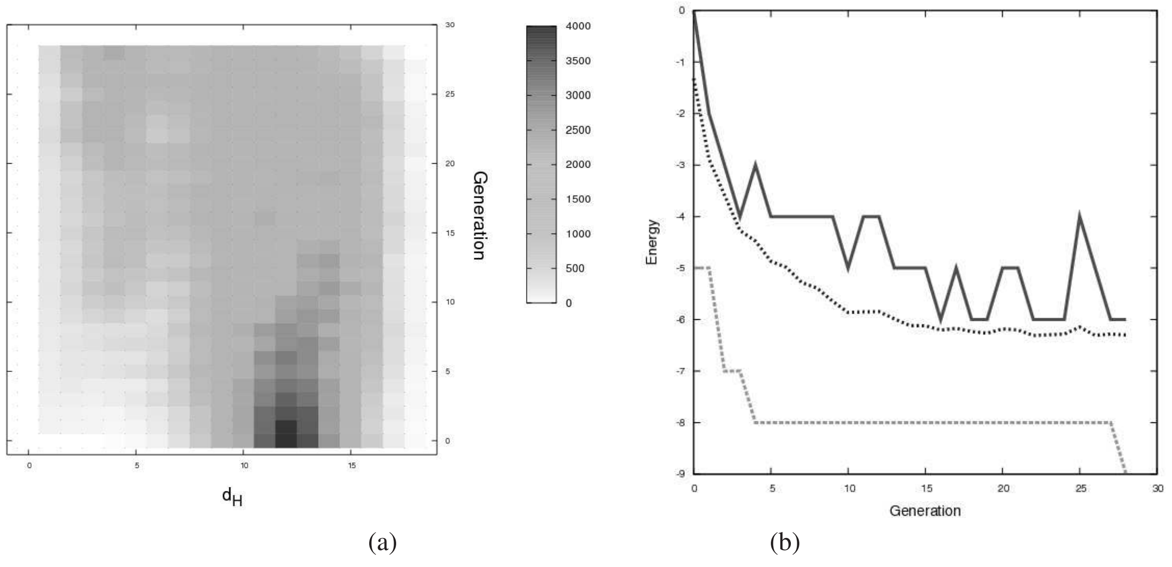


Figure 5: (a) The density of pairwise Hamming distances, d_H, between population members throughout the calculation. (b) The change in energy throughout the calculation, showing the best (dashed), worst (solid) and average (doted) energies in each generation.

population diversity drastically decreases during the final stages of the calculation, not just in the final generation. This confirms that the calculation has not discovered the global minimum by chance, but a directed search strategy has been employed.

5 Conclusions

Although implementation of a modified constructor for use in the mutation phase of the IA has not always given greater success rates (especially for more challenging sequences), it has allowed for a more efficient search to be performed in some cases, showing a descrease in the number of fitness evaluation performed. The use of population diversity tracking allows a greater understanding of the algorithm's ability to explore areas of the potential energy surface of these simple model proteins. Areas of favourable local structure along the chain can be assessed, illustrating the important allele combinations that give rise to the determination of global minima. We are currently applying these approaches to more realistic protein models.

Acknowledgements

AJB thanks Dr Benjamin Curley for programming assistance and the University of Birmingham for PhD funding. Calculations were performed on the University of Birmingham's BlueBEAR 1500+ processor cluster, funded by the EPSRC (under SRIF3) and the University of Birmingham.

References

[1] K.M. Merz (ed.). *The Protein Folding problem and Structure Prediction*. Birkhauser, 1994.

[2] N. Krasnogor, W.E. Hart, J. Smith, and D. Pelta. Protein Structure Prediction with Evolutionary Algorithms. In *Proc. 1999 International Genetic and Evolutionary Computation Conference (GECCO99)*, Orlando, Florida, USA, 1999, pages 1569–1601.

[3] G.A. Cox and R.L. Johnston. Analyzing Energy Landscapes for Folding Model Proteins. *J. Chem. Phys.*, 124(20):204714, 2006.

[4] T. Beutler and K. Dill. A Fast Conformational Search Strategy for Finding Low Energy Structures of Model Proteins. *Protein Sci.* 5(10):2037–2043, 1996.

[5] R. Unger and J. Moult. Genetic Algorithms for Protein Folding Simulations. *J. Mol. Biol.*, 231:75–81, 1993.

[6] R. Ramakrishnan, B. Ramachandran, and J.F. Pekny. A Dynamic Monte Carlo Algorithm for Exploration of Dense Conformational Spaces in Heteropolymers. *J. Chem. Phys.*, 106(6):2418–2425, 1997.

[7] S. Kirkpatrick and C.D. Gellat. Optimization by Simulated Annealing. *Science*, 220(4598):671–680, 1983.

[8] W. Liu and B. Schmidt. Mapping of Genetic Algorithms for Protein Folding onto Computational Grids. *IEICE T. Inf. Syst.*, 89(2):589–596, 2006.

[9] J. Song, J. Cheng, T. Zeng, and J. Mau. A Novel Genetic Algorithm for HP Model Protein Folding. In *Proc. Sixth International Conference on Parallel and Distributed Computing Applications and Technologies (PDCAT'05)*, Dalian, China, 2005, pages 935–937.

[10] A. Shmygelska and H.H. Hoos. An Improved Ant Colony Optimization Algorithm for the 2D HP Protein Folding Problem. *Lect. Notes Comput. Sci.*, 2671:400–412, 2003.

[11] V. Cutello, G. Niscosia, M. Pavone, and J. Timmis. An Immune Algorithm for Protein Structure Prediction on Lattice Models. *IEEE T. Evol. Comput.*, 11(1):101–117, 2007.

[12] W.E. Hart, S. Istrail. HP Benchmarks. `www.cs.sandia.gov/tech_reports/compbio/tortilla-hp-benchmarks.html`.

[13] L.N. de Castro and J. Timmis. *Artifical Immune Systems and Their Applications*. Springer-Verlag, 1999.

APC Semantics for Petri Nets

Slavomír Šimoňák, Štefan Hudák and Štefan Korečko
Department of Computers and Informatics,
Technical University of Košice, Letná 9, 042 00 Košice, Slovak Republic
E-mail: slavomir.simonak@tuke.sk, stefan.hudak@tuke.sk, stefan.korecko@tuke.sk

Keywords: Petri nets semantics, process algebra, algebra of process components

Received: July 13, 2007

The paper deals with an algebraic semantics for Petri nets, based on a process algebra APC (Algebra of Process Components) by the authors. APC is tailored especially for describing processes in Petri nets. This is done by assigning special variables (called E-variables here) to every place of given Petri net, expressing processes initiated in those places. Algebraic semantics is then given as a parallel composition of all the variables, whose corresponding places hold token(s) within the initial marking. Resulting algebraic specification preserves operational behavior of the original net-based specification.

Povzetek: Članek opisuje algebro semantike za Petri mreže.

1 Introduction

An assertion widely accepted in formal methods community states, that there will never be invented a single formal method, that will cover all aspects of the system in acceptable way [12]. The latter is mainly because of the complexity of the system and vast variety of its features (aspects) to be covered for the system to be modeled and designed. As a consequence of that situation, many formal description techniques (FDTs) exist and are used nowadays. That reflects the fact that one feature f of the system is more readily expressed in FDT (say) F_1, than it is the case for f in F_2. To cope with that situation there have been attempts to integrate two or more formal methods. Main motivation for using FDT in design and analysis of computer-based systems lies in everyday growing dependence of human society on such the systems, particulary those applied in safety-critical domains (such as military weaponry, aircraft transport, medicine, etc.). The situation just described put strong requirements on new methods of the design, analysing and maintaining of such the systems. Time-critical issue of to be able to cope with malicious behaviour of the systems in limited time period, dictates strongly to deal with the problem in an automated way. The latter is impossible without using FDTs in proper combination and integration in the frame of computer-based design and analysis environments. Yet a formal way how to incorporate conditions to guarantee the safety of system designed is an example of another problem which underlines the importance of expolitation of FDTs. Guided by the considerations mentioned an approach has been applied at the home institution of the authors to create an environment for the design and analysis of discrete systems based on integration of three FDTs: Petri Nets (PN), process algebras and B-Method. That is why we have chosen the acronym mFDTE for it (multi FDT Environment). The choice of the FDTs has been motivated by their abilities to cover in some mutually complementary ways a chosen set of system's features. In this work we pay attention to two of the FDTs chosen: PN and process algebras. While on one side PN posses nice properties suited for system modelling (formal and graphical language) and analysis (invariants, reachability), on the other side they suffer from a lack of formally sound and effective methods of their de/composition. The latter can be considered as an essential drawback as far as the modular system design is concerned. Process algebras (CCS, CSP, ACP)[1] on the other hand support composionality, by their definition, so PN and process algebras can be considered to meet the complementarity property in the above sense.

The paper is organized in the following way: Section 1 is introductory one, whole related works are briefly summarized in Section 2. In Section 3, basic notions and definitions for the class of Petri nets used are given. Algebra of Process Components is defined in Section 4. Notion of term is presented as a mean for describing processes, axioms are given and operational semantics is assigned to process expressions (terms). Section 5 concentrates on the algebraic semantics construction for a Petri net given. A special variable is assigned to every place of the Petri net. Construction rules are defined for assigning a term to the variable which represents all the computations which can be initiated at the corresponding place. An example provided in Section 6 demonstrates the approach introduced above. Section 7 concludes the paper and contains a summary of the results and concepts presented.

2 Related work

An active research has been performed in the area of combining Petri nets and process algebras during last years

[4, 5, 2, 6, 10, 11]. It will be mentioned further [3], where authors propose an approach to algebraic semantics for Hierarchical P/T nets. PTNA (Place/Transition Net Algebra) is defined there, based on process algebra ACP and an algebraic semantics for P/T nets is given such that a P/T net and its term representation have the same operational behavior. The actions of the algebra presented correspond to the consumption and production of tokens by transitions. Results achieved are further extended to hierarchical P/T nets. In [11] relations among nets, terms and formulas are treated. Particularly relations are defined via properly defined semantics: net semantics of terms and process semantics of nets. The most influential works in the line we follow here are [5, 2]. In [5] relations between the process algebra, called there PBC (Petri Box Calculus) and a class of Place Transition Nets (safe P/T nets) are studied. Syntax and semantics of PBC terms are carefully selected to allow to define a transformation yielding P/T nets preserving structural operational semantics of the source terms. The transformation allows composition of P/T nets. In work [2] authors treat the issue of partial-order algebras and their relations to P/T nets based on the theory of BPA and ACP.

Within our work we propose the general approach to characterising PN in form of E-(B-) terms. General (not only interleaving) semantics is given, and the results obtained in this respect are published in [9]. mFDT Environment is under construction, based on FDT interfaces by the authors, which aims to integrate the three formal methods mentioned above.

3 Petri Nets

We assume the class of ordinary Petri nets [18] within this paper, and brief description of the basic notions follows [7].

Definition 1. *The Petri net is a 4-tuple $N = (P, T, pre, post)$, where P is a finite set of places, T is a finite set of transitions ($P \cap T = \emptyset$), pre: $P \times T \to \{0, 1\}$ is the preset function and post: $P \times T \to \{0, 1\}$ is the postset function.*

By the marking of PN $N = (P, T, pre, post)$ we mean a totally defined function m: $P \to \mathbb{N}$, where $\mathbb{N}$ is the set of natural numbers. We denote marked net with initial marking m_0 as $N_0 = (N, m_0)$ or $N_0 = (P, T, pre, post, m_0)$.

Some useful notations can be defined:

$$^{\bullet}t = \{p | pre(p,t) \neq 0\} \text{the set of preconditions of } t$$

$$t^{\bullet} = \{p | post(p,t) \neq 0\} \text{the set of postconditions of } t$$

$$p^{\bullet} = \{t | pre(p,t) \neq 0\},\ ^{\bullet}p = \{t | post(p,t) \neq 0\}$$

We say that t is enabled in m (and denote it $m \xrightarrow{t}$) if for every $p \in {}^{\bullet}t, m(p) \geq pre(p,t)$. The effect of firing t in m is the creation of a new marking m' ($m \xrightarrow{t} m'$) and m' is defined in the following way:

$$m'(p) = m(p) - pre(p,t) + post(p,t), p \in P, t \in T$$

Denotation $(N, m) \xrightarrow{t} (N, m')$ is alternatively used within the paper for expressing a step of computation ($m \xrightarrow{t} m'$) within the Petri net N. The set of reachable markings for given Petri net $N_0 = (P, T, pre, post, m_0)$ we define by

$$\mathcal{R}(N_0) = \{m | m_0 \xrightarrow{\sigma} m\}$$

where $\sigma = t_1, t_2 ... t_r$ stands for an admissible firing sequence in N_0. We also can define the language of Petri net N_0:

$$\mathcal{L}(N_0) = \{\sigma \in T^* | m_0 \xrightarrow{\sigma} m\}$$

4 APC - Algebra of Process Components

Process algebra APC [8, 16] is inspired by the process algebra ACP [1]. ACP is modified in a way, that allows for comfortable description of PN processes in the algebraic way. We use the same operators for the sequential ($\cdot$) and the alternative (+) composition respectively and corresponding axioms also hold in algebra APC (Table 1). ACP's communication function (γ) and its extension - communication merge operator ($|$), are not present in APC. A composition function (π) and a special composition operator ($|||$) are introduced into the algebra APC instead. The impact of an introduction of the two operators will be treated later.

APC is defined as a couple (P, Σ), where P (the domain) is represented by the set of constants, set of variables and set of all processes (terms) we are able to express. Σ (the signature) contains function (operator) symbols. It is supposed that, the set of variables contain arbitrary many of them (x,y, ...). Terms containing variable(s) are called *open terms*, otherwise terms are *closed.*

4.1 Syntactical issues

From the syntactical point of view APC contains a number of constants $a, b, c, ...$ (we use the set $A = \{a, b, c, ...\}$ for referring to them) a special constant δ (deadlock) and operators: +, $\cdot$, $\|$ (parallel composition), $⌊\!⌊$ (left merge) and $|||$ (process component composition). It also contains a (partial) commutative composition function π, denoting the merging of process components. Now we will define APC terms:

Definition 2. *1. variables are APC terms,*

2. constants $a \in A$ and a special constant δ are APC terms,

3. if u, v are APC terms, then $u + v, u \cdot v$, $u \parallel v, u ⌊\!⌊ v, u|||v$ are APC terms,

4. if u is APC term, then $u^{[c]}$, $c \in \mathbb{N}$ is also APC term.

All these terms are part of P - the domain of APC. P can further be subdivided into two parts P_A and P_C ($P = P_A \cup P_C$). Terms belonging to the set P_A defined by items 1, 2 and 3 of Definition 2 (i.e. those without the superscript notation) represent the set of *true processes*. Terms from the set P_C (superscripted) represent the set of *process components*.

Only difference between the (true) process and process component is, that while the process is able to execute actions, process component is introduced for synchronization purpose only. The latter is only able to join with its counterpart(s) (other process component(s) fitting for being synchronized to) in order to form a true process. The composition function π is defined as follows:

$$\pi : P_C \times ... \times P_C \rightarrow P_A \tag{1}$$

A connection between the composition function π and the process component composition operator ($|||$) can be expressed as: $x_1|||...|||x_n = \pi(x_1, ..., x_n)$ when $\pi(x_1, ..., x_n)$ is defined. Axioms of algebra APC can be found in Table 1. Within the table $u, v, z, x_1, ..., x_n$ stand for processes, $a \in A$ and δ are constants.

$u + v = v + u$	A1
$u + u = u$	A2
$(u + v) + z = u + (v + z)$	A3
$(u + v) \cdot z = uz + vz$	A4
$(u \cdot v) \cdot z = u \cdot (v \cdot z)$	A5
$u\|v = u \lfloor\!\lfloor v + v \lfloor\!\lfloor u + u|||v$	A6
$(u + v)|||z = u|||z + v|||z$	A7
$u|||(v + z) = u|||v + u|||z$	A8
$au \lfloor\!\lfloor v = a(u\|v)$	A9
$(u + v) \lfloor\!\lfloor z = u \lfloor\!\lfloor z + v \lfloor\!\lfloor z$	A10
$x_1|||...|||x_n = \pi(x_1, ..., x_n)$ if $\pi(x_1, ..., x_n)$ is defined $x_1|||...|||x_n = \delta$ otherwise	A11
$u + \delta = u$	A12
$\delta \cdot u = \delta$	A13

Table 1: Axioms of APC

Theorem 1. *In the case of the parallel composition of more than two processes the following equality (expansion theorem) can be proven:*

$$x_1\|...\|x_n = \sum_{1\le i\le n} x_i \lfloor\!\lfloor (\mathop{\|}\limits^{1\le j\le n, j\ne i} x_j) + \tag{2}$$

$$\sum_{2\le k\le n} \sum^{1\le i_1<...<i_k\le n} (x_{i_1}|||...|||x_{i_k}) \lfloor\!\lfloor (\mathop{\|}\limits^{1\le j\le n, j\ne i_1,...,i_k} x_j)$$

Proof 1. *By induction on n, the number of processes. The case for $n = 2$ is treated by the axiom A6, Table 1. The induction step is as follows:*

$$(x_1\|...\|x_{n+1}) = (x_1\|...\|x_n)\|x_{n+1}$$

Denoting the RHS of original theorem as E, we can write:

$$(x_1\|...\|x_n)\|x_{n+1} = E \lfloor\!\lfloor x_{n+1} + x_{n+1} \lfloor\!\lfloor E + E|||x_{n+1} \tag{3}$$

Now we have three summands, each of them will be treated separately. Let's start dealing with the first of them.

$$E \lfloor\!\lfloor x_{n+1} = (\sum_{1\le i\le n} x_i \lfloor\!\lfloor (\mathop{\|}\limits^{1\le j\le n, j\ne i} x_j)) \lfloor\!\lfloor x_{n+1} +$$

$$(\sum_{2\le k\le n} \sum^{1\le i_1<...<i_k\le n} (x_{i_1}|||...|||x_{i_k})$$

$$\lfloor\!\lfloor (\mathop{\|}\limits^{1\le j\le n, j\ne i_1,...,i_k} x_j)) \lfloor\!\lfloor x_{n+1} =$$

$$\sum_{1\le i\le n} x_i \lfloor\!\lfloor ((\mathop{\|}\limits^{1\le j\le n, j\ne i} x_j) \lfloor\!\lfloor x_{n+1}) +$$

$$\sum_{2\le k\le n} \sum^{1\le i_1<...<i_k\le n} (x_{i_1}|||...|||x_{i_k})$$

$$\lfloor\!\lfloor ((\mathop{\|}\limits^{1\le j\le n, j\ne i_1,...,i_k} x_j) \lfloor\!\lfloor x_{n+1}) =$$

$$\sum_{1\le i\le n} x_i \lfloor\!\lfloor (\mathop{\|}\limits^{1\le j\le n+1, j\ne i} x_j) +$$

$$\sum_{2\le k\le n} \sum^{1\le i_1<...<i_k\le n} (x_{i_1}|||...|||x_{i_k})$$

$$\lfloor\!\lfloor (\mathop{\|}\limits^{1\le j\le n+1, j\ne i_1,...,i_k} x_j)$$

The second summand of (3) can be expressed as follows:

$$x_{n+1} \lfloor\!\lfloor E = x_{n+1} \lfloor\!\lfloor (\mathop{\|}\limits^{1\le j\le n+1, j\ne n+1} x_j)$$

The third one represents the process components composition:

$$E|||x_{n+1} = (\sum_{1\le i\le n} x_i \lfloor\!\lfloor (\mathop{\|}\limits^{1\le j\le n, j\ne i} x_j))|||x_{n+1} +$$

$$(\sum_{2\le k\le n} \sum^{1\le i_1<...<i_k\le n} (x_{i_1}|||...|||x_{i_k})$$

$$\lfloor\!\lfloor (\mathop{\|}\limits^{1\le j\le n, j\ne i_1,...,i_k} x_j))|||x_{n+1} =$$

$$\sum_{1\le i\le n} (x_i|||x_{n+1}) \lfloor\!\lfloor (\mathop{\|}\limits^{1\le j\le n, j\ne i} x_j) +$$

$$(\sum_{3\le k\le n+1} \sum^{1\le i_1<...<i_k\le n+1} (x_{i_1}|||...|||x_{i_k})$$

$$\lfloor\!\lfloor (\mathop{\|}\limits^{1\le j\le n, j\ne i_1,...,i_k} x_j))$$

Summing up the three summands we have:

$$(x_1\|...\|x_n\|x_{n+1}) = \sum_{1\leq i\leq n+1} x_i \mathbin{\lfloor\!\lfloor} (\mathop{\|}_{1\leq j\leq n+1, j\neq i} x_j) + \sum_{2\leq k\leq n+1} \sum_{1\leq i_1<...<i_k\leq n+1} (x_{i_1}|||...|||x_{i_k}) \mathbin{\lfloor\!\lfloor} (\mathop{\|}_{1\leq j\leq n+1, j\neq i_1,...,i_k} x_j)$$

□

4.2 Semantics issues

Constants $a, b, c \in A$ are called atomic actions, and are considered indivisible actions (events). The sequential composition operator $(\cdot)$ function can be explained as follows: $x\cdot y$ is the process that first executes x and after finishing it, starts y. The alternative composition $(+)$: $x+y$ is the process that either executes x or y (choice). The meaning of parallel composition $(\|)$ follows: considering the merge of two processes $x\|y$, we recognize three possibilities to proceed. Either we start with a first step of x (given by $x\mathbin{\lfloor\!\lfloor} y$), or a first step from y ($y\mathbin{\lfloor\!\lfloor} x$) or we check a possibility to compose processes by means of process components composition operator - $x|||y$. The result of this composition of course is different from δ only in a case, when the composition function π is defined.

To assign an operational semantics to process expressions, we determine, which actions a process can perform. The fact, that process represented by the term t can execute action a and turn to the term s is denoted by: $t \xrightarrow{a} s$ (or alternatively a is enabled in t). The symbol $\surd$ stands for successful termination and thus $t \xrightarrow{a} \surd$ denotes a fact that t can terminate by executing a. An inductive definition of action relations is given in the Table 2.

$a \xrightarrow{a} \surd$	
$u \xrightarrow{a} u' \Rightarrow$	$u+v \xrightarrow{a} u'$
	$v+u \xrightarrow{a} u'$
	$u\cdot v \xrightarrow{a} u'\cdot v$
	$u\|v \xrightarrow{a} u'\|v$
	$v\|u \xrightarrow{a} v\|u'$
	$u\mathbin{\lfloor\!\lfloor} v \xrightarrow{a} u'\|v$
$u \xrightarrow{a} u'$, $\pi(u^{[1]},...,u^{[n]}) = u \Rightarrow u^{[1]}|||...|||u^{[n]} \xrightarrow{a} u'$	
$a\in A,\ u\in P_A,\ u^{[1]},...,u^{[n]} \in P_c,\ u',v\in P$	

Table 2: Transition relations for APC terms

5 APC semantics for Petri Nets

In this section the transformation description is given in detail [16]. We start with creating a special variable for every place in the PN N to be transformed. We call these variables *E-variables* here, and they will be bound to terms, representing possible computations started from given place in PN, later. So the value (term) assigned to a particular variable depends on the structure of the net in the vicinity of a place associated. So considering the place p, variable $E(p)$ will be bound to a term representing all the computations within the net N, which are initiated in p.

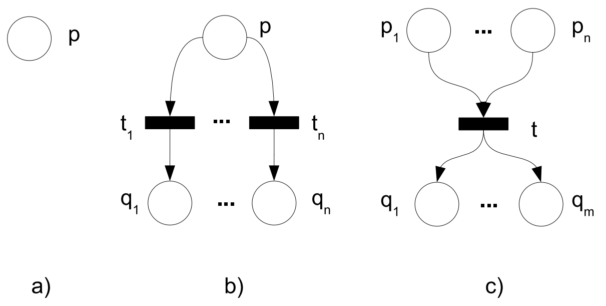


Figure 1: Petri net fragments

Basic situations are captured in Fig. 1. In the case a) a situation is depicted, where no arcs are connected to the place investigated.

This results to the assignment of a term representing no computations to the variable corresponding to such place, i.e. δ (deadlock). Case b) stands for an alternative composition (choice). If a token is situated in place p, a choice is to be made, and only one of transitions $t_1, ..., t_n$ can fire. Case c) represents general composition, where tokens must be present in all pre-places of transition t. If some of these places does not contain a token, firing of t is not possible. After firing of t, however post-places of it are marked and thus processes initiated in those places are enabled.

General composition (case c) can be understood as a generalization of the three basic compositions - sequential, parallel and synchronization (Fig. 2) - three of four basic composition mechanisms (with alternative composition) used within the APC. If n is the number of pre-places and m, the number of post-places of a transition t:

- $n = 1 \wedge m = 1$ we obtain sequential composition (case a) of Fig. 2),
- $n = 1 \wedge m > 1$ we obtain parallel composition (case b) of Fig. 2),
- $n > 1 \wedge m = 1$ we obtain synchronization (case c) of Fig. 2).

Now we can proceed by constructing terms representing possible computations for given places of PN N. These will be bound to a corresponding E-variables in a way given by the definition:

Definition 3. *According to the structure of Petri net in the vicinity of a given place, terms are bound to corresponding variables for elementary situations depicted in figures Fig. 1 and Fig. 2 as follows:*

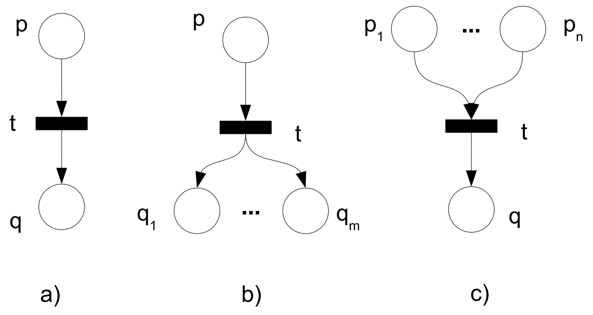


Figure 2: Basic compositions as a special cases of the general composition

- *deadlock (Fig. 1a):* $E(p) = \delta$
- *alternative composition (Fig. 1b):* $E(p) = t_1 \cdot E(q_1) + t_2 \cdot E(q_2) + ... + t_n \cdot E(q_n)$
- *sequential composition (Fig. 2a):* $E(p) = t \cdot E(q)$
- *parallel composition (Fig. 2b):* $E(p) = t \cdot (E(q_1) \parallel ... \parallel E(q_n))$
- *synchronization (Fig. 2c):* $E(p_1) = (t \cdot E(q))^{[1]}, E(p_2) = (t \cdot E(q))^{[2]}, ..., E(p_n) = (t \cdot E(q))^{[n]}$,
  and the composition function is defined:
  $\pi((t \cdot E(q))^{[1]}, ..., (t \cdot E(q))^{[n]}) = t \cdot E(q)$ *or* $\pi(E(p_1), ..., E(p_n)) = t \cdot E(q)$
- *general composition (Fig. 1c):* $E(p_1) = (t \cdot (E(q_1) \parallel ... \parallel E(q_m)))^{[1]}$,
  $E(p_2) = (t \cdot (E(q_1) \parallel ... \parallel E(q_m)))^{[2]}, ...,$
  $E(p_n) = (t \cdot (E(q_1) \parallel ... \parallel E(q_m)))^{[n]}$,
  and the composition function is defined in the following way:
  $\pi(E(p_1), ..., E(p_n)) = t \cdot (E(q_1) \parallel ... \parallel E(q_m))$
- *transition without post-place(s) (Fig. 3a):* $E(p) = t$
- *transition without pre-place(s) (Fig. 3b): a new place is added, such that firing properties of a transition given are preserved (Fig. 3c):* $E(p) = t \cdot E(q)$.

In the case of the synchronization we can observe that, all variables composed are assigned to terms representing process components instead of true processes. These components, if all are present within the term representing the net computation, can merge together by means of composition function π, and form the true process (able to execute action t and then to behave like $E(q)$).

Taking into account the case, when a transition without pre-places occurs within the net structure (Fig. 3b), the following solution is proposed: for every such transition t, a new pre-place is added, such that the firing properties of transition t are preserved (Fig. 3c). This is achieved by setting the initial marking of given place to ω, where

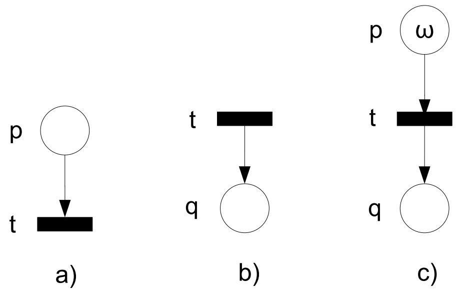


Figure 3: Transitions without input/output

$\forall n \in \mathbb{N} : \omega \pm n = \omega$. In fact, this causes the transition t can be fired infinitely many times. Combining these basic principles, we are able to construct terms for more complicated net structures.

Definition 4. *Let the PN N is given by $N = (P, T, pre, post)$, $m \in \mathbb{N}^k$ stands for its initial marking, and $k = |P|$. Then the APC semantics for N with marking m is given by the formula:*

$$\mathcal{A}(N, m) = E(p_1)^{(i_1)} \| ... \| E(p_k)^{(i_k)} \tag{4}$$

Within the Definition 4, $E(p_i)$ stands for an APC-term defined according to PN structure in the vicinity of the place p_i (Definition 3). The value i_j is given by $i_j = m(p_j)$, so it represents the marking with respect to place $p_j, 1 \le j \le k$. $E(p_j)^{(i)}$ is defined as a term $E(p_j) \| ... \| E(p_j)$, and represents a multiple (i-times) parallel composition of a process $E(p_j)$. Note that $E(p_j)^{(0)} = \delta$. When $i_j = \omega$ for place p_j, it means that $E(p_j)$ can occur infinitely many times in resulting composition.

Theorem 2. *For given PN $N = (P, T, pre, post)$, APC-term p, representing an algebraic (APC) semantics for the net N, transition $t \in T$ and m, m' markings of N, following implication holds:*

$$(N, m) \xrightarrow{t} (N, m') \Rightarrow \mathcal{A}(N, m) \xrightarrow{t} \mathcal{A}(N, m')$$

Proof 2. *The proof is given by the induction on a structure of the net. Let us suppose, that a step in computation of N exists: $(N, m) \xrightarrow{t} (N, m')$, then*

$$\forall p_i \in ({}^\bullet t) : m(p_i) \ge pre(p_i, t)$$

$$\forall p_i \in P : m'(p_i) = m(p_i) - pre(p_i, t) + post(p_i, t)$$

Algebraic semantics for Petri net N with marking m is given by:

$$\mathcal{A}(N, m) = E(p_1)^{(i_1)} \| ... \| E(p_k)^{(i_k)}, \ k = |P| \tag{5}$$

Transition t fired in N within a step can be of two kinds:

1. $|{}^\bullet t| = 1$
2. $|{}^\bullet t| \ge 2$

According to the transition relations of APC (Table 2), a step can be made by executing the action of the true process or by merging the process components together with execution of an action associated. Let us explore the two cases:

1. *If $|^\bullet t| = 1$ is the case, the situation is captured in the Fig. 4, case a). Then within the Petri net N holds: $m(p_a) \geq pre(p_a, t)$ (so the place p_a contains a token(s)) and also $m'(p_i) = m(p_i) + post(p_i, t) - pre(p_i, t), p_i \in P$ is a new marking after firing of the transition t. If $E(p_a) = t \cdot (E(p_c) \| ... \| E(p_d))$ is corresponding APC semantics for process initiated in the place p_a, then a step (action t) is enabled $E(p_a) \xrightarrow{t} E(p_c) \| ... \| E(p_d)$, since $E(p_a)$ is present in specification $\mathcal{A}(N, m)$. Let values $j_l, l \in \{1, ..., k\}$ are given by: $j_l = i_l - pre(p_l, t) + post(p_l, t)$. When a step $\mathcal{A}(N, m) \xrightarrow{t} \mathcal{A}(N, m')$ occurs, corresponding APC semantics of the net (N, m') is given by:*

$$\mathcal{A}(N, m') = E(p_1)^{(j_1)} \| ... \| E(p_k)^{(j_k)}, \quad k = |P| \tag{6}$$

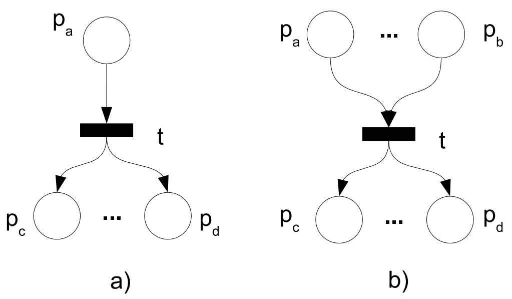


Figure 4: Two cases considered for a step in computation of N

2. *Here transition t firing in Petri net N occurs, for which $|^\bullet t| \geq 2$ holds. Situation is depicted in Fig. 4, case b). Within the Petri net N there holds: $m(p_a) \geq pre(p_a, t), \ldots, m(p_b) \geq pre(p_b, t)$ (so the places $p_a, \ldots, p_b$ contain enough tokens) and thus transition t can fire. From the definition of APC semantics for Petri net N (5) and Definition 3, we have that a step from (N, m) is represented by:*

$$E(p_a)^{(i_a)} \| ... \| E(p_b)^{(i_b)} \xrightarrow{t} E(p_c)^{(j_c)} \| ... \| E(p_d)^{(j_d)} \tag{7}$$

The step is enabled in $\mathcal{A}(N, m)$ since values $i_a, \ldots, i_b$ are given by number of tokens in corresponding places. According to the definition, variables $E(p_a)$,...,$E(p_b)$ are bound to process components and composition function is defined: $\pi(E(p_a), ..., E(p_b)) = t \cdot (E(p_c) \| ... \| E(p_d))$. The step (7) thus is enabled, and (6) holds, where: $j_1 = i_1 + post(p_1, t) - pre(p_1, t)$,...,$j_k = i_k + post(p_k, t) - pre(p_k, t)$.

We can conclude, that if a step in Petri net N with marking m is enabled, so it is enabled also in corresponding algebraic representation $\mathcal{A}(N, m)$.

□

We give a small example here, representing the configuration of Petri net N sometimes called confusion (Fig. 5) for the sake of clarity.

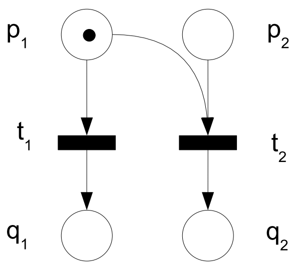


Figure 5: Confusion

First, APC-terms are assigned to variables created for every place of Petri net N.

$$E(p_1) = t_1 \cdot E(q_1) + (t_2 \cdot E(q_2))^{[1]}, \\ E(p_2) = (t_2 \cdot E(q_2))^{[2]}$$

Next, the composition function π is defined:

$$\pi((t_2 \cdot E(q_2))^{[1]}, (t_2 \cdot E(q_2))^{[2]}) = t_2 \cdot E(q_2) \tag{8}$$

Within the initial marking m_0 of Petri net N, the only place holding a token is p_1, so only the variable corresponding to this place will be included within the equation describing the algebraic semantics of N.

$$\mathcal{A}(N, m_0) = E(p_1) = t_1 \cdot E(q_1) + (t_2 \cdot E(q_2))^{[1]} = \\ t_1 \cdot E(q_1) + \delta = t_1 \cdot E(q_1)$$

In the configuration depicted, the only transition enabled is t_1 and it can fire. This is not the case of the transition t_2, because the place p_2, the pre-place of t_2 is not marked. The same could be observed within the APC representation - only one process component $((t_2 \cdot E(q_2))^{[1]})$ is present within the equation (so the mapping π cannot be used to produce a true process) and it thus cannot perform any action and is replaced by the δ.

6 An example

An example has been chosen to demonstrate a way how the transformation rules proposed can be used. The Petri net N (depicted in Fig. 6) represents a synchronization problem for sharing one resource by two processes. System modeled consists of two processes (let the first process represented by the places p_1, p_2 and transitions t_1, t_3, be named A, and the second one (p_3, p_4, t_2, t_4) be named B). The resource shared is represented by the place p_0. The token at this place indicates the resource is free to be shared either by process A or process B. The place p_2 stands for the condition 'process A is using the resource', firing transition t_1 starts the resource usage and firing t_3 ends it. Similarly, for process B, firing t_2 starts and firing t_4 ends the usage respectively. A token occurrence in the place p_3 indicates the resource is used by the process B.

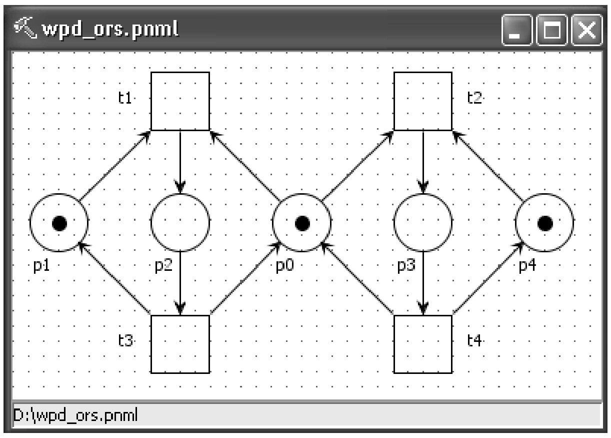


Figure 6: Petri net for resource sharing

We start with assigning APC-terms to variables created for every place of PN N, according to the structure of the net in the vicinity of the corresponding place.

$$E(p_1) = (t_1 \cdot E(p_2))^{[1]},\ E(p_2) = t_3(E(p_0) \parallel E(p_1)),$$
$$E(p_3) = t_4(E(p_0) \parallel E(p_4)),\ E(p_4) = (t_2 \cdot E(p_3))^{[1]},$$
$$E(p_0) = (t_1 \cdot E(p_2))^{[2]} + (t_2 \cdot E(p_3))^{[2]}$$

Composition function π is defined in two cases:

$$\pi((t_1 \cdot E(p_2))^{[1]}, (t_1 \cdot E(p_2))^{[2]}) = t_1 \cdot E(p_2) \qquad (9)$$

$$\pi((t_2 \cdot E(p_3))^{[1]}, (t_2 \cdot E(p_3))^{[2]}) = t_2 \cdot E(p_3) \qquad (10)$$

Since the initial marking of Petri net N is given as $m_0 = (1, 1, 0, 0, 1)$, only three places (p_0, p_1 and p_4) hold tokens, and only variables corresponding to these places will take place in equation describing the algebraic semantics of PN N.

$$\mathcal{A}(N, m_0) = E(p_0) \parallel E(p_1) \parallel E(p_4) \qquad (11)$$

Since the term on the RHS of equation (11) represents parallel composition of three processes, we expand it according to (2):

$$= E(p_0) \mathbin{\|\!\lfloor} (E(p_1) \parallel E(p_4)) + E(p_1) \mathbin{\|\!\lfloor} (E(p_0) \parallel E(p_4)) + E(p_4) \mathbin{\|\!\lfloor} (E(p_0) \parallel E(p_1)) +$$
$$(E(p_0) ||| E(p_1)) \mathbin{\|\!\lfloor} E(p_4) + (E(p_0) ||| E(p_4)) \mathbin{\|\!\lfloor} E(p_1) + (E(p_1) ||| E(p_4)) \mathbin{\|\!\lfloor} E(p_0) + (E(p_0) ||| E(p_1) ||| E(p_4))$$

After substituting terms assigned (bound) to variables $E(p_0)$, $E(p_1)$ and $E(p_4)$, we can write:

$$= [(t_1 \cdot E(p_2))^{[2]} + (t_2 \cdot E(p_3))^{[2]}] \mathbin{\|\!\lfloor} ((t_1 \cdot E(p_2))^{[1]} \parallel (t_2 \cdot E(p_3))^{[1]}) +$$
$$(t_1 \cdot E(p_2))^{[1]} \mathbin{\|\!\lfloor} [((t_1 \cdot E(p_2))^{[2]} + (t_2 \cdot E(p_3))^{[2]}] \parallel (t_2 \cdot E(p_3))^{[1]}) +$$
$$(t_2 \cdot E(p_3))^{[1]} \mathbin{\|\!\lfloor} [((t_1 \cdot E(p_2))^{[2]} + (t_2 \cdot E(p_3))^{[2]}] \parallel (t_1 \cdot E(p_2))^{[1]}) + ([(t_1 \cdot E(p_2))^{[2]} + (t_2 \cdot E(p_3))^{[2]}] ||| (t_1 \cdot E(p_2))^{[1]}) \mathbin{\|\!\lfloor} (t_2 \cdot E(p_3))^{[1]} + ([(t_1 \cdot E(p_2))^{[2]} + (t_2 \cdot E(p_3))^{[2]}] ||| (t_2 \cdot E(p_3))^{[1]}) \mathbin{\|\!\lfloor} (t_1 \cdot E(p_2))^{[1]} +$$
$$((t_1 \cdot E(p_2))^{[1]} ||| (t_2 \cdot E(p_3))^{[1]}) \mathbin{\|\!\lfloor} [(t_1 \cdot E(p_2))^{[2]} + (t_2 \cdot E(p_3))^{[2]}] + ([(t_1 \cdot E(p_2))^{[2]} + (t_2 \cdot E(p_3))^{[2]}] ||| (t_1 \cdot E(p_2))^{[1]} ||| (t_2 \cdot E(p_3))^{[1]})$$

Using the composition (π) definition (9, 10) and axioms associated (A11, A13), we have:

$$(t_1 \cdot E(p_2)) \mathbin{\|\!\lfloor} (t_2 \cdot E(p_3))^{[1]} + (t_2 \cdot E(p_3)) \mathbin{\|\!\lfloor} (t_1 \cdot E(p_2))^{[1]}$$

Using the left merge operator axiom (A9) and substituting for $E(p_2)$ and $E(p_3)$ we obtain:

$$= t_1(E(p_2) \parallel (t_2 \cdot E(p_3))^{[1]}) + t_2(E(p_3) \parallel (t_1 \cdot E(p_2))^{[1]})$$
$$= t_1(t_3(E(p_0) \parallel E(p_1)) \parallel (t_2 \cdot E(p_3))^{[1]}) + t_2(t_4(E(p_0) \parallel E(p_4)) \parallel (t_1 \cdot E(p_2))^{[1]})$$

Considering all the cases for two processes composed by the parallel composition operator ($\|$) and using the axiom A6:

$$= t_1(t_3(E(p_0) \parallel E(p_1)) \mathbin{\|\!\lfloor} (t_2 \cdot E(p_3))^{[1]} + (t_2 \cdot E(p_3))^{[1]} \mathbin{\|\!\lfloor} t_3(E(p_0) \parallel E(p_1)) + t_3(E(p_0) \parallel E(p_1)) ||| (t_2 \cdot E(p_3))^{[1]}) +$$
$$t_2(t_4(E(p_0) \parallel E(p_4)) \mathbin{\|\!\lfloor} (t_1 \cdot E(p_2))^{[1]} + (t_1 \cdot E(p_2))^{[1]} \mathbin{\|\!\lfloor} t_4(E(p_0) \parallel E(p_4)) + t_4(E(p_0) \parallel E(p_4)) ||| (t_1 \cdot E(p_2))^{[1]})$$
$$= t_1(t_3(E(p_0) \parallel E(p_1)) \mathbin{\|\!\lfloor} (t_2 \cdot E(p_3))^{[1]} + \delta + \delta) + t_2(t_4(E(p_0) \parallel E(p_4)) \mathbin{\|\!\lfloor} (t_1 \cdot E(p_2))^{[1]} + \delta + \delta)$$
$$= t_1 t_3((E(p_0) \parallel E(p_1)) \parallel (t_2 \cdot E(p_3))^{[1]}) + t_2 t_4((E(p_0) \parallel E(p_4)) \parallel (t_1 \cdot E(p_2))^{[1]})$$

Since $E(p_4) = (t_2 \cdot E(p_3))^{[1]}$ and $E(p_1) = (t_1 \cdot E(p_2))^{[1]}$, we can write:

$$= t_1 t_3((E(p_0) \parallel E(p_1)) \parallel E(p_4)) + t_2 t_4((E(p_0) \parallel E(p_4)) \parallel E(p_1))$$
$$= t_1 t_3(E(p_0) \parallel E(p_1) \parallel E(p_4)) + t_2 t_4(E(p_0) \parallel E(p_1) \parallel E(p_4))$$

Using axiom A4, the term becomes even simpler:

$$= (t_1 t_3 + t_2 t_4)(E(p_0) \parallel E(p_1) \parallel E(p_4))$$

Here we can observe a parallel composition of the three variables, from which we started our derivation ($E(p_0) \parallel E(p_1) \parallel E(p_4)$). In terms of Petri nets, the initial marking was reached again. Prefix $(t_1t_3+t_2t_4)$ represents the traces of processes. The sequential composition operator is often omitted, so we can state that the APC semantics is finally given by the following equation:

$$\mathcal{A}(N, m_0) = (t_1t_3 + t_2t_4)^{\omega}$$

7 Conclusion

In this paper a general method was presented for constructing an algebraic semantics of Petri nets, based on Algebra of Process Components (APC) by the authors. The notion of process component is introduced in order to model synchronization, which, in case of Petri nets, is modeled in a natural way. A variable is created for every place of given net and a term is bound to this variable, which express the process initiated in the corresponding place. The description of process representing computations of Petri net is given by the parallel composition of all the variables associated with places holding token(s) within the initial marking. Traces of processes can be observed in addition to changes on the PN marking along the computation. Resulting algebraic specification can further be analyzed using process algebra tools like CWB-NC, etc. A proof of identical operational behavior has also been provided.

The PETRI2APC tool, a practical implementation of method presented, is intended to be a part of multi FDT (mFDT) environment - an environment for designing and analysing of discrete systems based on three formal methods with useful complementary properties. The methods considered are Petri nets, Process algebra and B-Method. The mFDT environment is under development at DCI FEEI TU of Košice.

Acknowledgement

This work is supported in part by the VEGA 1/3140/06 and in part by the VEGA 1/4073/07 grant projects of Slovak Grant Agency.

References

[1] Baeten, J.C.M., Weijland, W.P.: *Process Algebra*, Cambridge University Press, ISBN 0 521 40043 0, pp.248, 1990.

[2] Baeten, J.C.M., Basten, T.: *Partial-Order Process Algebra*, in Handbook of Process Algebra, Elsevier Science, Amsterdam, The Netherlands, 2000, 78pp.

[3] Basten, T., Voorhoeve, M.: *An Algebraic Semantics for Hierarchical P/T Nets*, Computing Science Report, Eindhoven University of Technology, pp.32, 1995.

[4] Best, E.: *Semantics of Sequential and Parallel Programs*, Prentice Hall Europe, ISBN 0-13-460643-4, pp.352, 1996.

[5] Best, E., Devillers,R., Koutny,M.: *Petri Nets, Process Algebras and Concurrent Programming Languages* , Lecture on Petri Nets II:Applications, LNCS , Advances in Petri Nets (W.Reisig,G.Rozenberg Eds.), Springer , Berlin 1998, ISBN 3-540-65306-6,pp.1-84.

[6] Desel, J., Juhás, G., Lorenz, R.: *Process Semantics of Petri Nets over Partial Algebra*, Proceedings of ICATPN 2000, 21-st International Conference on Application and Theory of Petri Nets, vol.1825 of LNCS, pp.146-165, Springer-Verlag, 2000.

[7] Hudák, Š.: *Reachability Analysis of Systems Based on Petri Nets*, Elfa s.r.o. Košice, ISBN 80-88964-07-5, pp.272, 1999.

[8] Hudák, Š., Šimoňák, S.: *APC - Algebra of Process Components*, Proceedings of EMES'03, Oradea, Felix Spa, Romania, May 2003, pp. 57-63, ISSN 1223 - 2106.

[9] Hudák Š., Šimoňák S., Korečko Š., N.Kovacs A.: Formal Specifications and de/compositional approach to design and analysis of discrete systems, Computer Science and Technology Research Survey, Košice, Elfa, pp. 8-46, 2007.

[10] Mayr, R.: *Combining Petri nets and PA-processes*, in Theoretical Aspects of Computer Software (TACS'97), volume 1281 of Lecture Notes in Computer Science, pages 547–561, Sendai, Japan, 1997. Springer Verlag.

[11] Olderog, E.R.: *Nets, Terms and Formulas*, ISBN 0 521 40044 9, Cambridge University Press, 1991.

[12] Paige, R.F.: *Formal Method Integration via Heterogeneous Notations*, PhD Dissertation, November 1997.

[13] *PNML–Petri Net Markup Language*, URL: http://www.informatik.hu-berlin.de/ top/pnml/

[14] *PNML Framework*, URL: http://www-src.lip6.fr/logiciels/mars/PNML/

[15] Šimoňák, S., Hudák, Š., Korečko, Š.: *ACP2PETRI: a tool for FDT integration support*, Proceedings of 8th International Conference EMES'05, Oradea, Felix Spa, Romania, 2005, pp. 122-127, ISSN 1223-2106.

[16] Šimoňák, S.: *Formal methods integration based on Petri nets and process algebra transformations*, PhD Dissertation, DCI FEEI TU Košice, 2003. (In Slovak)

[17] Starke, P.H.: *Processes in Petri Nets*, LNCS 117, Fundamentals of Computation Theory, ISBN 3-540-10854-8, Springer-Verlag, 1981.

[18] *Petri Nets World (A Classification of PN)*, URL: http://www.informatik.uni-hamburg.de/TGI/PetriNets/classification/

Dynamic Slicing of Aspect-Oriented Programs

Durga Prasad Mohapatra
Department of CSE
National Institute of Technology
Rourkela-769008, India
E-mail: durga@nitrkl.ac.in

Madhusmita Sahu
Department of MCA
C V Raman Computer Academy
Bhubaneswar-752054,India
E-mail: madhu_sahu@yahoo.com

Rajeev Kumar and Rajib Mall
Department of CSE
Indian Institute of Technology
Kharagpur-721302, India
E-mail: {rkumar, rajib}@cse.iitkgp.ernet.in

Keywords: program slice, aspect-oriented programming, AspectJ, dynamic aspect-oriented dependence graph (DADG), dynamic dependence slicing tool (DDST), trace file based dynamic slicing (TBDS) algorithm

Received: July 2, 2007

Program slicing is a decomposition technique which has many applications in various software engineering activities such as program debugging, testing, maintenance etc. Aspect-oriented programming (AOP) is a new programming paradigm that enables modular implementation of cross-cutting concerns such as exception handling, security, synchronization, logging etc. The unique features of AOP such as join-point, advice, aspect, introduction *etc. pose difficulties for slicing of AOPs. We propose a dynamic slicing algorithm for aspect-oriented programs. Our algorithm uses a dependence-based representation called Dynamic Aspect-Oriented Dependence Graph (DADG) as the intermediate program representation. The DADG is an arc-classified digraph which represents various dynamic dependences between the statements of the aspect-oriented program. We have used a trace file to store the execution history of the program. We have developed a tool called* Dynamic Depenedence Slicing Tool (DDST) *to implement our algorithm. We have tested our algorithm on many programs for 40-50 runs. The resulting dynamic slice is precise as we create a node in the DADG for each occurrence of the statement in the execution trace.*

Povzetek: Opisana je modularna gradnja objektno orientiranih programov.

1 Introduction

The concept of a program slice was first introduced by Weiser [26]. Program slicing [38] is a decomposition technique which extracts program statements related to a particular computation from a program. A program slice is constructed with respect to a slicing criterion. A *slicing criterion* is a tuple $< s, v >$ where s is a statement in a program and v is a variable used or defined at s. A program slice can be *static* or *dynamic*. A *static* slice consists of all statements of a program that might affect the value of a variable at a program point of interest for *every possible* inputs to the program. In contrast, a *dynamic* slice consists of only those statements that actually affect the value of a variable at a program point of interest for a *particular* set of inputs to the program.

Aspect-oriented programming (AOP) is a new programming paradigm that enables modular implementation of cross-cutting concerns [17] such as exception handling, security, synchronization, logging. This concept was proposed by Gregor Kiczales et al. [16]. Expressing such cross-cutting concerns using standard language constructs produces poorly structured code since these concerns are tangled with the basic functionality of the code. This increases the system complexity and makes maintenance considerably more difficult.

AOP [2, 3] attempts to solve this problem by allowing the programmer to develop cross-cutting concerns as full stand-alone modules called *aspects*. The main idea behind AOP is to construct a program by describing each concern separately.

Aspect-oriented programming languages present unique opportunities and problems for program analysis schemes. For example, to perform program slicing on aspect-oriented software, specific aspect-oriented features such as *join-point*, *advice*, *aspect*, *introduction* must be handled appropriately. Although these features provide great strengths to model the cross-cutting concerns in an aspect-oriented program, they introduce difficulties to analyze the program.

A major aim of any slicing technique is to realize as small a slice with respect to a slicing criterion as possible since smaller slices are found to be more useful for different applications. Much of the literature on program slicing is concerned with improving the algorithms for slicing in terms of reducing the size of the slice and improving the efficiency of the slicing algorithm. Now-a-days, many programs are aspect-oriented. These aspect-oriented programs are quite large and complex. It is much difficult to debug and test these products. Program slicing techniques have been found to be useful in applications such as program understanding, debugging, testing, software maintenance and reverse engineering etc. [12, 27, 29, 31]. Particularly dynamic program slicing is used in interactive applications such as debugging and testing of programs. Therefore the dynamic slicing techniques need to be efficient. This requires to develop efficient slicing algorithms as well as suitable intermediate representations for aspect-oriented programs. Researchers have developed many representations for procedural and object-oriented programs [11, 21, 24, 27, 28, 29, 33, 39], but very few work has been carried out for representation of aspect-oriented programs [22, 25]. Due to the specific features of aspect-oriented programming language, existing slicing algorithms for procedural or object-oriented programming languages cannot be applied directly to aspect-oriented programs. Therefore, there is a pressing necessity to devise suitable intermediate representations and efficient algorithms for dynamic slicing of aspect-oriented programs.

With this motivation for developing techniques for dynamic slicing of aspect-oriented programs, we identify the following objective. The main objective of our research work is to develop an efficient dynamic slicing algorithm. To address this broad objective, we identify the following goals:

– to develop a suitable intermediate representation for aspect-oriented programs on which the slicing algorithm can be applied.

– to develop a dynamic slicing algorithm for aspect-oriented programs, using the proposed intermediate representation.

In this paper, we propose a new intermediate representation for aspect-oriented programs. We call this representation as **Dynamic Aspect-Oriented Dependence Graph (DADG)**. Then, we propose a dynamic slicing algorithm for aspect-oriented programs. We have used a trace file to store the execution history of the source code. So, we have named our algorithm *Trace file Based Dynamic Slicing* (TBDS) algorithm. Our algorithm computes *precise* dynamic slices as we create a node in the DADG for each occurrence of the statement in the execution trace.

The rest of the paper is organized as follows. In Section 2, we discuss some related works. In Section 3, we present a brief introduction to Aspect-Oriented Programming (AOP). In Section 4, we describe some notions of dynamic slices of aspect-oriented programs. Section 5 discusses the dynamic aspect-oriented dependence graph (DADG) for aspect-oriented programs and also describes the construction of DADG. In Section 6, we discuss the computation of dynamic slices of aspect-oriented programs using DADG. In Section 7, we present the implementation details of our work. Section 8 concludes the paper.

2 Related work

Horwitz et al. [33] developed a system dependence graph (SDG) as an intermediate program representation for procedural programs with multiple procedures. They proposed a two-phase graph reachability algorithm on the SDG to compute inter-procedural slice. The slice consists of the union of vertices marked in both the phases.

Later, Larsen and Harrold [24] extended the SDG of Horwitz et al. [33] to represent object-oriented programs. Their [24] extended SDG incorporates many object-oriented features such as classes, objects, inheritance, polymorphism etc. After constructing the SDG, Larsen and Harrold [24] used the two-phase algorithm to compute the static slice of an object-oriented program. Later, Liang and Harrold [11] developed a more efficient intermediate representation of object-oriented programs which is an extension to the SDG of Larsen and Harrold [24]. Their [11] SDG represents objects that are used as parameters or data members in other objects, the effects of polymorphism on parameters and parameter bindings. The data members for different objects can be distinguished using this approach. Later many researchers have extended the work on static slicing of object-oriented programs [11, 39]. But they [11, 39] have not considered the aspect-oriented features.

Also dynamic slicing of OOPs have been addressed by several researchers [21, 27, 28, 29]. Korel and Laski [7] introduced the concept of dynamic program slicing. Agrawal and Horgan [19] presented the first algorithm for finding dynamic slices of procedural programs using dependence graphs.

Zhao [21] extended the *dynamic dependence graph* (DDG) of Agarwal and Horgan [18] for the representation of various dynamic dependences between statement instances for a particular execution of an object-oriented program. Zhao [21] named this graph *dynamic object-oriented dependence graph* (DODG). He used a two-phase algorithm on the DODG for the computation of dynamic slices of object-oriented programs.

Song et al. [36] proposed a method to compute forward dynamic slices of object-oriented programs using *dynamic object relationship diagram* (DORD). They computed the dynamic slices for each statement immediately after the statement is executed. The dynamic slices of all executed statements have been obtained after the execution of the last statement.

Xu et al. [9] extended their earlier work [39] to dynamically slice object-oriented programs. Their method uses *object program dependence graph* (OPDG) and other static information to reduce the information to be traced during execution and computes dynamic slices combining static dependence information and dynamic execution of the program.

Wang et al. [37] proposed a new algorithm for dynamic slicing of Java programs which operates on compressed bytecode traces. According to their approach, first, the bytecode stream corresponding to an execution trace of a Java program is compactly represented. Then, a backward traversal of the compressed program trace is performed to compute data/control dependences on-the-fly. The slice is updated as these dependences are encountered during trace traversal.

Mohapatra et al. [29, 30] have developed edge-marking and node-marking dynamic slicing techniques for object-oriented programs. Their algorithms are based on marking and unmarking the edges (nodes) of the graph appropriately, as and when dependences arise and cease. Many researchers [8, 21, 28, 31] have extended the work on dynamic slicing of object-oriented programs. But, none of the researchers [8, 21, 28, 29, 30, 31] have considered the aspect-oriented features.

Zhao [22] was the first to develop the aspect-oriented system dependence graph (ASDG) to represent aspect-oriented programs. The ASDG is constructed by combining the SDG for non-aspect code, the aspect dependence graph (ADG) for aspect code and some additional dependence arcs used to connect the SDG and ADG. Then, Zhao [22] used the two-phase slicing algorithm proposed by Larsen and Harrold [24] to compute static slice of aspect-oriented programs.

Braak [34] extended the ASDG proposed by Zhao [22, 23] to include inter-type declarations in the graph. Each inter-type declaration was represented in the form of a field or a method as a successor of the particular class. Then, Braak [34] used the two-phase slicing algorithm of Horwitz et al. [33] to find the static slice of an aspect-oriented program. Braak [34] has not addressed the dynamic slicing aspects.

We have proposed an approach for computation of dynamic slice using a dependence graph based intermediate representation called *dynamic aspect-oriented dependence graph* (DADG). We have used a trace file to store the execution history of the aspect-oriented program. Our DADG correctly represents the aspect-oriented features such as pointcuts, advices etc. Also, weaving process is correctly represented in the DADG. The TBDS algorithm computes *precise* dynamic slices as we create separate vertices in the DADG for each occurrence of the statement in the execution trace.

3 Aspect-oriented programming

In this section, we first discuss the basic concepts of aspect-oriented programming. Then, we briefly describe AspectJ: an aspect-oriented programming language. Next, we present some features of AspectJ.

3.1 Basic concepts

Gregor Kiczales et al. [16] introduced the concept of Aspect-Oriented Programming (AOP) at Xerox Palo Alto Research Center (PARC) in 1996. An *aspect* is an *area of concern* that cuts across the structure of a program. *Concern* is defined as some functionality or requirement necessary in a system, which has been implemented in a code structure [4, 6, 17, 35]. Examples of aspects are data storage, user interface, platform-specific code, security, distribution, logging, class structure, threading etc.

The strength of aspect-oriented programming is the enabling of a better separation of concerns, by allowing the programmer to create cross-cutting concerns as program modules. *Cross-cutting concerns* are those parts, or aspects, of the program that are scattered across multiple program modules, and tangled with other modules in standard design.

```
void transfer(Account fromAccount, Account toAccount, int amount){
  if (!getCurrentUser().canPerform(OP_TRANSFER)){
    throw new SecurityException();
  }
  if (amount<0){
    throw new NegativeTransferException();
  }
  if (fromAccount.getBalance()<amount){
    throw new InsufficientFundsException();
  }
  Transaction tx=database.newTransaction();
  try{
    fromAccount.withdraw(amount);
    toAccount.deposit(amount);
    tx.commit();
    systemLog.logOperation(OP_TRANSFER,fromAccount,toAccount,amount);
  }
  catch(Exception e){
    tx.rollback();
  }
}
```

Figure 1: An example program

Let us consider the example program given in Figure 1. The objective of this program is to transfer an amount from one account to another in a banking application. In this example, various cross-cutting concerns such as transactions, security, logging etc. are tangled with the basic functionality (sometimes called as the *business logic concern*). If there is a need to change the security considerations for the application, then it would require a major effort since security-related operations appear scattered across numerous methods. This means that the cross-cutting concerns do not get properly encapsulated in their own modules and this increases the system complexity.

The goal of aspect-oriented programming (AOP) is to make it possible to deal with cross-cutting aspects of a system's behavior as separately as possible. Although the hierarchical modularity of object-oriented languages is extremely useful, they are inherently unable to modularize cross-cutting concerns in complex systems. Aspect-oriented programming provides language mechanisms to explicitly capture the cross-cutting structure.

To better support the expression of cross-cutting design decisions, AOP uses a component language to describe the basic functionality of the system and aspect languages to describe the different cross-cutting properties. The components and the aspects are then combined into a system using an *aspect weaver* [10]. The *aspect weaver* makes it possible for an *advice* to be activated at appropriate *join points* during run-time. Thus, a source code is modified by inserting aspect-specific statements at join points.

3.2 AspectJ: an aspect-oriented programming language

Several different Aspect-oriented programming systems have been built, including AML (Aspect Markup Language), an environment for sparse matrix computation [20], RG (Reverse Graphics), an environment for creating image processing systems [5] etc. The most popular AOP language is AspectJ. An AspectJ program is divided into two parts: base code or non-aspect code and aspect code. The *base code* includes classes, interfaces and other standard Java constructs. The *aspect code* implements the cross-cutting concerns in the program. Other aspect-oriented frameworks include COOL (COOrdination Language) for expressing synchronization concerns [13], RIDL (Remote Invocation Data transfer Language) for expressing distribution concerns [13], JBOSS, Spring AOP, AspectWerkz [10, 15] etc.

AspectJ was created by Chris Maeda [16] at Xerox Palo Alto Research Center (PARC). This is an aspect-oriented extension to Java programming language. In other words, we can say that AspectJ is compatible with current Java platform [14]. There are four types of compatibility:

- *Upward compatibility*- all legal Java programs must be legal AspectJ programs.

- *Platform compatibility*- all legal AspectJ programs must run on standard Java virtual machines.

- *Tool compatibility*- it must be possible to extend existing tools to support AspectJ in a natural way; this includes IDEs, documentation tools, and design tools.

- *Programmer compatibility*- Programming with AspectJ must feel like a natural extension of programming with Java.

3.3 Features of AspectJ

AspectJ adds some new features to Java. These features include *join points*, *pointcut*, *advice*, *aspect*, *introduction* or *inter-type declaration*. We explain these features below.

- *Join Points*- These are well-defined points in the execution of a program, such as, method call (a point where method is called), method execution (a point where method is invoked) and method reception join points (a point where a method received a call, but this method is not executed yet).

- *Pointcut*- This is a means of referring to collections of join points and certain values at those join points. AspectJ defines several primitive *pointcut designators* that can identify all types of join points. For example, in Figure 2, the pointcut *factorialOperation* at statement 13 picks out join points i.e. the pointcut *factorialOperation* picks out each call to the method *factorial()* of an instance of the class *TestFactorial*, where an *int* is being passed as an argument and it makes the value of that argument to be available to the enclosing advice or pointcut.

- *Advice*- It is a method-like construct which is used to define cross-cutting behavior at join points. This is used to define some code that is executed when a pointcut is reached. Advice brings together a pointcut (to pick out join points) and a body of code (to run at each of those join points). There are three types of advice in AspectJ: *after*, *before*, *around*.

  (i) *After*- *After advice* on a particular join point runs after the program proceeds with that join point. For example, *after* advice on a method call join point runs after the method body has run, just before control is returned to the caller. For example, in Figure 2, the *after* advice at statement 16 runs just after each join point picked out by the pointcut *factorialOperation* and before the control is returned to the calling method.

  (ii) *Before*- *Before* advice runs as a join point is reached, before the program proceeds with the join point. For example, *before advice* on a method call join point runs before the actual method starts running, just after the arguments to the method call are evaluated. For example, in Figure 2, the *before* advice at statement 14 runs just before the join points picked out by the pointcut *factorialOperation*.

(iii) *Around- Around advice* on a join point runs as the join point is reached, and has explicit control over whether the program proceeds with the join point.

Additionally, there are two special cases of after advice: *after returning* and *after throwing*, corresponding to the two ways a sub-computation can return through a join point.

(i) *After returning- After returning* advice runs just after each join point picked out by the pointcut, but only if it returns normally. The return value can be accessed. After the advice runs, the return value is returned. For example, in Figure 2, the *after returning* advice at statement 16 runs just after each join point picked out by the pointcut *factorialOperation*, but only if it returns normally. The return value can be accessed and it is named *result* in Figure 2 at statement 16. After the advice runs, the return value is returned.

(ii) *After throwing- After throwing* advice runs just after each join point picked out by the pointcut, but only when it throws an exception. The advice re-raises the exception after it is done.

– *Aspect-* These are units of modular cross-cutting implementations composed of pointcuts, advices, and ordinary JAVA member declarations. An *aspect* is a cross-cutting type, defined by the aspect declaration. Aspects are defined by *aspect* declarations, which have a similar form of class declarations. For example, in Figure 2, there is one aspect named *OptimizeFactorialAspect* at statement 12.

– *Introduction* or *Inter-Type Declaration-* It allows an aspect to add methods, fields or interfaces to existing classes. It can be *public* or *private*. An introduction declared as *private* can be referred or accessed *only* by the code in the aspect that declared it. An introduction declared as *public* can be accessed by *any* code.

– *Pointcut Designator-* It is a formula that specifies the set of join points to which a piece of advice is applicable. A pointcut designator identifies all types of join points. A pointcut designator simply matches certain join points at runtime. For example, in Figure 2, the pointcut designator

call (long TestFactorial.factorial(int))

at statement 13 matches all method calls to *factorial* from an instance of the class *TestFactorial*.

Pointcuts can be combined using logical operators *and* (&&), *or* (||) and *not* (!). For example, in Figure 2, the compound pointcut designator

call (long TestFactorial.factorial(int)) && args(n)

at statement 13 refers to all method calls to *factorial()* of an instance of *TestFactorial*, where the argument of type *int* is passed to the method *factorial()*.

User-defined pointcut designators are defined with *pointcut* declaration. For example, in Figure 2, the declaration

public pointcut factorialOperation(int n):
call (long TestFactorial.factorial(int)) && args(n)

at statement 13 defines a new pointcut designator, *factorialOperation*, that specifies a call to the method *factorial()* of an instance of *TestFactorial* and the argument passed to the method to be of type *int*.

For example, Figure 2 shows an AspectJ program for finding the factorial of a number. The program is divided into two parts: the base code or non-aspect code contains the class *TestFactorial* and the aspect code *OptimizeFactorialAspect* contains the advices and pointcuts. Any AspectJ implementation ensures that both the codes i.e., aspect code and base code run together in a properly coordinated fashion. Such type of process is called *aspect weaving*. The key component for this process is *aspect-weaver* which makes the applicable advices to run at the appropriate join points.

4 Dynamic slicing of aspect-oriented programs

In this section, we present the basic concepts and the definitions which will be used in our algorithm.

Definition 1 (Digraph): A *digraph* is an ordered pair (V, A), where V is a finite set of elements called *vertices* and A is a finite set of elements called *edges* and $A \subseteq V \times V$.

Definition 2 (Arc-classified digraph): An *arc-classified digraph* is an n-tuple $(V, A_1, A_2, \ldots, A_{n-1})$ such that every (V, A_i), $(i = 1, 2, \ldots, n-1)$ is a digraph and $A_i \cap A_j = \emptyset$ for $i = 1, 2, \ldots, n-1$ and $j = 1, 2, \ldots, n-1$ and $i \neq j$.

Definition 3 (Path): A path from vertex u to vertex v in a digraph (V, A) is a sequence of vertices $u, i_1, i_2, \ldots, i_k, v$ in V such that $(u, i_1), (i_1, i_2), \ldots, (i_k, v)$ are edges in A.

Definition 4 (Flow graph): The *flow graph* of an aspect-oriented program is a quadruple $(V, A, Start, Stop)$ where (V, A) is a digraph, *Start* $\in V$ is a distinguished node of in-degree 0 called the *start node*, *Stop* $\in V$ is a distinguished node of out-degree 0 called the *stop node*, there is a path from *Start* to every other node in the graph, and there is a path from every other node in the graph to *Stop*.

Definition 5 (Control flow graph(CFG)): Let the set

```
    import java.util.*;
    public class TestFactorial{
        private static int n;
1:      public static void main(String[] args){
2:      n=Integer.parseInt(args[0]);
3:      System.out.println("Result:  "+factorial(n)+"\n");
       }

4:     public static long factorial(int n){
       long p;
5:     if(n>0){
6:        p=1;
7:        while(n>0){
8:            p=p*n;
9:            n--;
         }
  }
        else
10:        p=1;
11:   return p;
       }
       }
```

Non−aspect Code (Base Code)

```
     import java.util.*;
12:   public aspect OptimizeFactorialAspect{
13:    public pointcut factorialOperation(int n):
            call(long TestFactorial.factorial(int)) && args(n);
14:     before(int n): factorialOperation(n){
15:        System.out.println("Seeking factorial for  "+n);
    }
16:    after(int n) returning (long result): factorialOperation(n){
17:       System.out.println("Getting the factorial for  "+n);
       }
       }
```

Aspect Code

Figure 2: An example AspectJ program

```
1 main()
2 {
3    int x, y, prod;
4     cin>> x;
5     cin>> y;
6     prod = 1;
7     while(x < 5) {
8        prod=prod*y;
9         ++ x; }
10    cout<< prod;
11    prod = y;
12    cout<< prod;
13  }
```

Figure 3: An example program

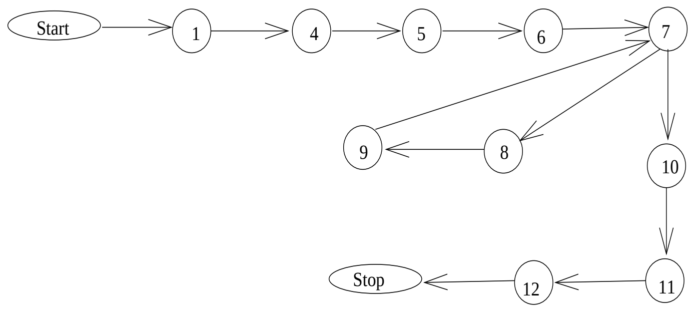


Figure 4: The CFG of the example program given in Figure 3

V represent the set of statements of a program P. The *control flow graph* of the program P is the flow graph G = $(V_1, A, Start, Stop)$ where $V_1 = V \cup \{Start, Stop\}$. An edge $(m, n) \in A$ indicates the possible flow of control from the node m to the node n. Note that the existence of an edge (x, y) in the control flow graph means that control *must* transfer from x to y during program execution. Figure 4 represents the CFG of the example program given in Figure 3.

Definition 6 (Dominance): If x and y are two nodes in a flow graph G then x dominates y iff every path from *Start* to y passes through x. y post-dominates x iff every path from x to *Stop* passes through y.

Let x and y be nodes in a flow graph G. Node x is said to be immediate post-dominator of node y iff x is a post-dominator of y, $x \neq y$ and each post-dominator $z \neq x$ of y post-dominates x. The post-dominator tree of a flow graph G is the tree that consists of the nodes of G, has the root Stop, and has an edge (x, y) iff x is the immediate post-dominator of y.

Consider the flow graph of the example program of Figure 3, which is given in Figure 4. In the flow graph, each of the nodes 4, 5 and 6 dominates 7. Node 8 does not dominate node 10. Node 10 post dominates each of the nodes 4, 5, 6, 7, 8 and 9. Node 9 post dominates node 8. Node 9 post dominates none of the nodes 4, 5, 6, 7, 10, 11 and 12. Node 6 is the immediate post dominator of node 5. Node 10 is the immediate post dominator of node 7.

Definition 7 (Execution trace): An *execution trace* is a path that has actually been executed for some input data.

For example, for the input data $argv[0]$ = 4, the order of execution of the statements of the program given in Figure 2 is 1, 2, 3, 13, 14, 15, 4, 5, 6, 7, 8, 9, 7, 8, 9, 7, 8, 9, 7, 8, 9, 7, 16, 17, 11. This *execution trace* is given in Figure 5.

Definition 8 (Def(var)): Let var be a variable in a class in the program P. A vertex u of the DADG of P is said to be a *Def(var)* vertex if u represents a definition (assignment) statement that defines the variable var.

In the DADG given in Figure 6, vertices 6 and 8 are the *Def(p)* vertices.

Definition 9 (DefSet(var)): The set *DefSet(var)* denotes the set of all *Def(var)* vertices.

In the DADG given in Figure 6, *DefSet(p)*={6, 8}.

Definition 10 (Use(var)): Let var be a variable in a class in the program P. A vertex u of the DADG of P is said to be a *Use(var)* vertex if u represents a statement that uses the variable var.

In the DADG given in Figure 6, vertices 8 and 11 are the *Use(p)* vertices.

Definition 11 (UseSet(var)): The set *UseSet(var)* denotes the set of all *Use(var)* vertices.

In the DADG given in Figure 6, *UseSet(p)*={8, 11}.

5 The dynamic aspect-oriented dependence graph (DADG)

In this section, we describe the definition and construction of the dynamic aspect-oriented dependence graph (DADG).

The DADG is an *arc-classified digraph* (V, A), where V is the set of vertices that correspond to the statements and predicates of the aspect-oriented programs, and A is the set of arcs between vertices in V representing dynamic dependence relationships that exist between statements. In the DADG of an aspect-oriented program, following types of dependence arcs may exist.

- control dependence arc
- data dependence arc
- weaving arc

Control dependences represent the control flow relationships of a program i.e., the predicates on which a statement or an expression depends during execution.

Data dependences represent the relevant data flow relationships of a program i.e., the flow of data between statements and expressions.

For example, in the DADG given in Figure 6, there is a data dependency between vertex 6 and 8, because vertex 6 is *Def(p)* vertex and vertex 8 is *Use(p)* vertex.

Weaving arcs reflect the joining of aspect code and non-aspect code at appropriate join points.

For example, in Figure 6 there is an weaving arc from vertex 13 to vertex 3 to connect vertex 13 to vertex 3 at the corresponding join point because, there is a function call at statement 3 and the corresponding pointcut at statement 13 captures that function call. Statement 14 represents a *before* advice. This means that the *advice* is executed before control flows to the corresponding function i.e., to the function *factorial()*. So, we add a *weaving arc* from vertex 4 to vertex 15. Similarly, statement 16 represents an *after* advice. This means that the *advice* is executed after the function *factorial()* has been executed and before control flows to the calling function i.e., the function *main()*. That's why we add a weaving arc from vertex 16 to vertex 7. After the execution of *after* advice at statement 17, control transfers to statement 11 where it returns a value i.e., the value of p to the calling function *main()*. So, a weaving arc is added from vertex 11 to vertex 17.

Our construction of dynamic aspect-oriented dependence graph of an aspect-oriented program is based on dynamic analysis of control flow and data flow of the program. The DADG of the program in Figure 2 corresponding to the execution trace in Figure 5 is given in Figure 6. In this figure, circles represent program statements, dotted lines represent data dependence arcs, solid lines represent control dependence arcs and dark dashed lines represent weaving arcs.

6 Computation of dynamic slices of aspect-oriented programs

Dynamic slicing of aspect-oriented programs is similar to that of object-oriented programs. However, due to the presence of pointcuts and advices, the tracing of dependences becomes much more complex.

Here, we formally define some notions of dynamic slicing of aspect-oriented programs. Let P be an aspect-oriented program and $G = (V, A)$ be the DADG of P. We compute the dynamic slice of an aspect-oriented program with respect to a slicing criterion.

- A *slicing criterion* for an aspect-oriented program is of the form $< p, q, e, n >$, where p is a statement, q is a variable used at p and e is an execution trace of the program with input n.
- A *dynamic slice* of an aspect-oriented program for a given slicing criterion $< p, q, e, n >$ consists of all the statements that have actually affected the value of the variable q at statement p.

Let DS_G be the dynamic slice of G on a given slicing criterion $< p, q, e, n >$. Then, DS_G is a subset of vertices of G i.e., $DS_G(p, q, e, n) \subseteq V$, such that for any $p' \in V$, $p' \in DS_G(p, q, e, n)$ if and only if there exists a path from p' to p in G. Since we have used a trace file to store the execution history of the aspect-oriented program, we have named our algorithm *Trace file Based Dynamic Slicing* (TBDS) algorithm for AOPs.

In this section, we present the TBDS algorithm in pseudo-code form to compute the dynamic slice of an aspect-oriented program.

Algorithm: TBDS algorithm

1. Creation of execution trace file: To create an execution trace file, do the following:

```
          public class TestFactorial
          private static int n;
1(1):     public static void main(String[ ] args)
2(1):     n=Integer.parseInt(args[0]);
3(1):     System.out.println("Result:  "+factorial(n)+"\n");
13(1):    public pointcut factorialOperation(int n): call(long TestFactorial.factorial(int)) && args(n);
14(1):    before(int n): factorialOperation(n)
15(1):    System.out.println("Seeking factorial for  "+n);
4(1):     public static long factorial(int n)
5(1):     if(n>0)
6(1):     p=1;
7(1):     while(n>0)
8(1):     p=p*n;
9(1):     n−−;
7(2):     while(n>0)
8(2):     p=p*n;
9(2):     n−−;
7(3):     while(n>0)
8(3):     p=p*n;
9(3):     n−−;
7(4):     while(n>0)
8(4):     p=p*n;
9(4):     n−−;
7(5):     while(n>0)
16(1):    after(int n) returning(long result): factorialOperation(n)
17(1):    System.out.println("Getting the factorial for  "+n);
11(1):    return p;
```

Figure 5: Execution trace of the program given in Figure 2 for argv[0] = 4

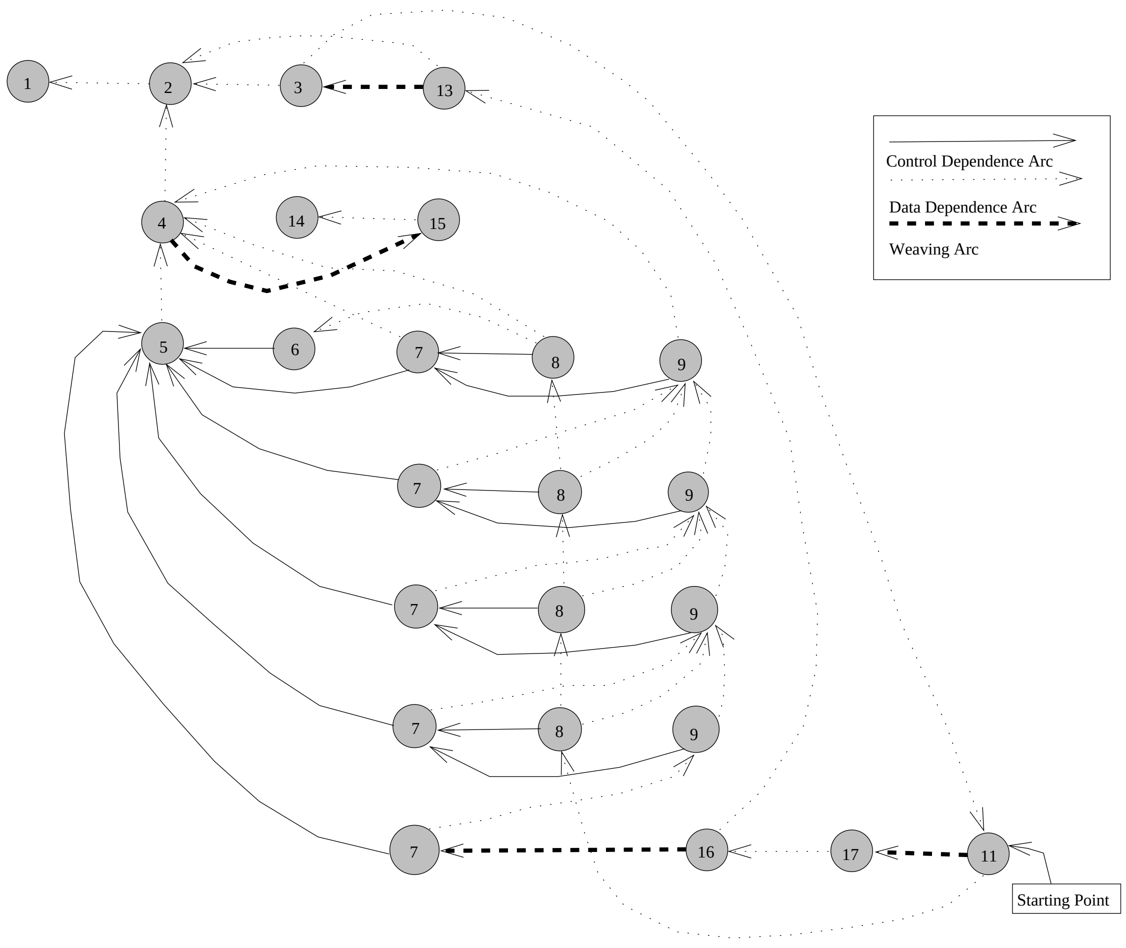


Figure 6: Dynamic aspect-oriented dependence graph for the execution trace given in Figure 5

   (a) For a given input, execute the program and store each statement s in the order of execution in a file after it has been executed.

   (b) If the program contains loops, then store each statement s inside the loop in the trace file after each time it has been executed.

2. Construction of DADG: To Construct the DADG of the aspect-oriented program P with respect to the trace file, do the following:

   (a) For each statement s in the trace file, create a vertex in the DADG.

   (b) For each occurrence of a statement s in the trace file, create a separate vertex.

   (c) Add all control dependence edges, data dependence edges and weaving edges to these vertices.

3. Computation of dynamic slice: To compute the dynamic slice over the DADG, do the following:

   (a) Perform the breadth-first or depth-first graph traversal over the DADG taking any vertex corresponding to the statement of interest as the starting point of traversal.

4. Mapping of the slice: To obtain the dynamic slice of the aspect-oriented program P, do the following:

   (a) Define a mapping function $f : DS_G(p, q, e, n) \rightarrow P$.

   (b) Map the resulting slice obtained in Step 3(a) over the DADG to the source code P using f since the slice may contain multiple occurrences of the same vertex.

Working of the algorithm: We illustrate the working of the TBDS algorithm with the help of an example. Consider the example AspectJ program given in Figure 2. Now, for the input data $argv[0]$ = 4, the program will execute the statements 1, 2, 3, 13, 14, 15, 4, 5, 6, 7, 8, 9, 7, 8, 9, 7, 8, 9, 7, 8, 9, 7, 16, 17, 11 in order. These statements are stored in a trace file. Figure 5 shows the corresponding execution trace file. Then, the *Dynamic Aspect-Oriented Dependence Graph* (DADG) is constructed with respect to this trace file in accordance with the step 2 of the TBDS algorithm. Figure 6 shows the DADG of the example program given in Figure 2 with respect to the trace file given in Figure 5. Since, for the input data $argv[0]$ = 4, the statements 8 and 9 are executed four times and statement 7 is executed five times, separate vertices are created for each occurrence of these statements.

Now, let us suppose that we have to compute the dynamic slice for the slicing criterion $< 11, p >$. Starting from the vertex 11, we can perform either the breadth-first search algorithm or depth-first search algorithm on the DADG. The breadth-first search algorithm yields the vertices 11, 17, 8, 16, 7, 8, 9, 7, 13, 5, 9, 7, 8, 9, 9, 3, 2, 4, 7, 8, 9, 1, 15, 7, 6, 14 and the depth-first search algorithm yields the vertices 11, 8, 9, 9, 9, 4, 15, 14, 2, 1, 7, 5, 7, 7, 8, 8, 8, 6, 7, 17, 16, 13, 3, 7, 9. The traversed vertices are shown as shaded vertices in Figure 6. Using the mapping function f, we can find the statements corresponding to these vertices. This gives us the required dynamic slice which is shown in rectangular boxes in Figure 7.

6.1 Complexity analysis

In the following, we discuss the space and time complexity of our DADG algorithm.

Space complexity: Let P be an aspect-oriented program and S be the length of execution of P. Each executed statement will be represented by a single vertex in the DADG. Thus, it can be stated that there are S number of vertices in the DADG corresponding to all executed statements of program P. Also, S numbers of statements are stored in the execution trace file. So, the space complexity of the trace file based algorithm is $O(S)$.

Time complexity: Let P be an aspect-oriented program and S be the length of execution of P. The total time complexity is due to four components:

1. time required to store each executed statement in a trace file which is $O(S)$.
2. time required to construct the DADG with respect to the execution trace file which is $O(S)$.
3. time required to traverse the DADG and to reach at the specified vertex which is $O(S^2)$.
4. time required to map the traversed vertices to source program P which is $O(S)$.

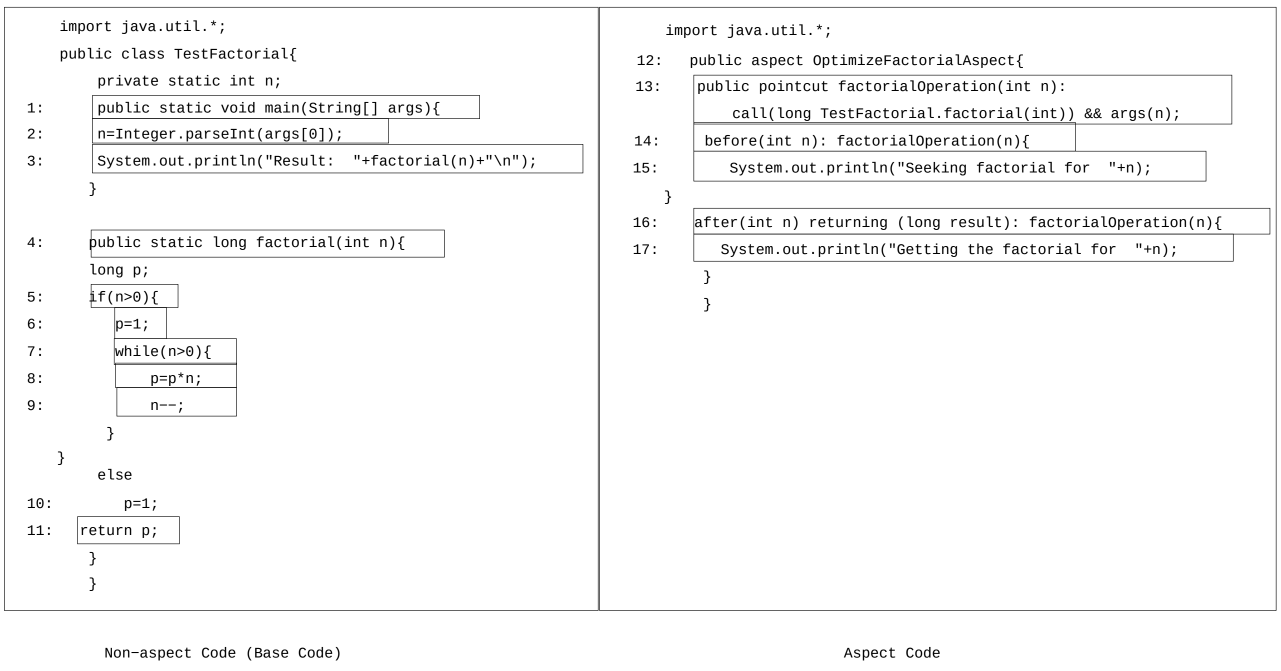
So, the time complexity of the trace file based algorithm is $O(S^2)$.

7 Implementation

In this section, we briefly describe the implementation of our algorithm. We have named our dynamic slicing tool *dynamic dependence slicing tool* (DDST) for aspect-oriented programs. First, we present an overview of our slicing tool and then, we discuss briefly the implementation of the slicing tool. Next we present some experimental results and then we compare our work with existing work.

7.1 Overview of DDST

The working of the slicing tool is schematically shown in Figure 8. The arrows in the figure show the data-flow among the different blocks of the tool. The blocks shown in rectangular boxes represent executable components and the blocks shown in ellipses represent passive components of the slicing tool.



```
     import java.util.*;
     public class TestFactorial{
          private static int n;
1:        public static void main(String[] args){
2:        n=Integer.parseInt(args[0]);
3:        System.out.println("Result:  "+factorial(n)+"\n");
         }

4:       public static long factorial(int n){
         long p;
5:       if(n>0){
6:          p=1;
7:          while(n>0){
8:              p=p*n;
9:              n--;
           }
     }
          else
10:          p=1;
11:    return p;
         }
         }
```

Non−aspect Code (Base Code)

```
     import java.util.*;
12:   public aspect OptimizeFactorialAspect{
13:    public pointcut factorialOperation(int n):
           call(long TestFactorial.factorial(int)) && args(n);
14:     before(int n): factorialOperation(n){
15:        System.out.println("Seeking factorial for  "+n);
    }
16:   after(int n) returning (long result): factorialOperation(n){
17:      System.out.println("Getting the factorial for  "+n);
        }
        }
```

Aspect Code

Figure 7: The dynamic slice of the program given in Figure 2 for the slicing criterion (11,p)

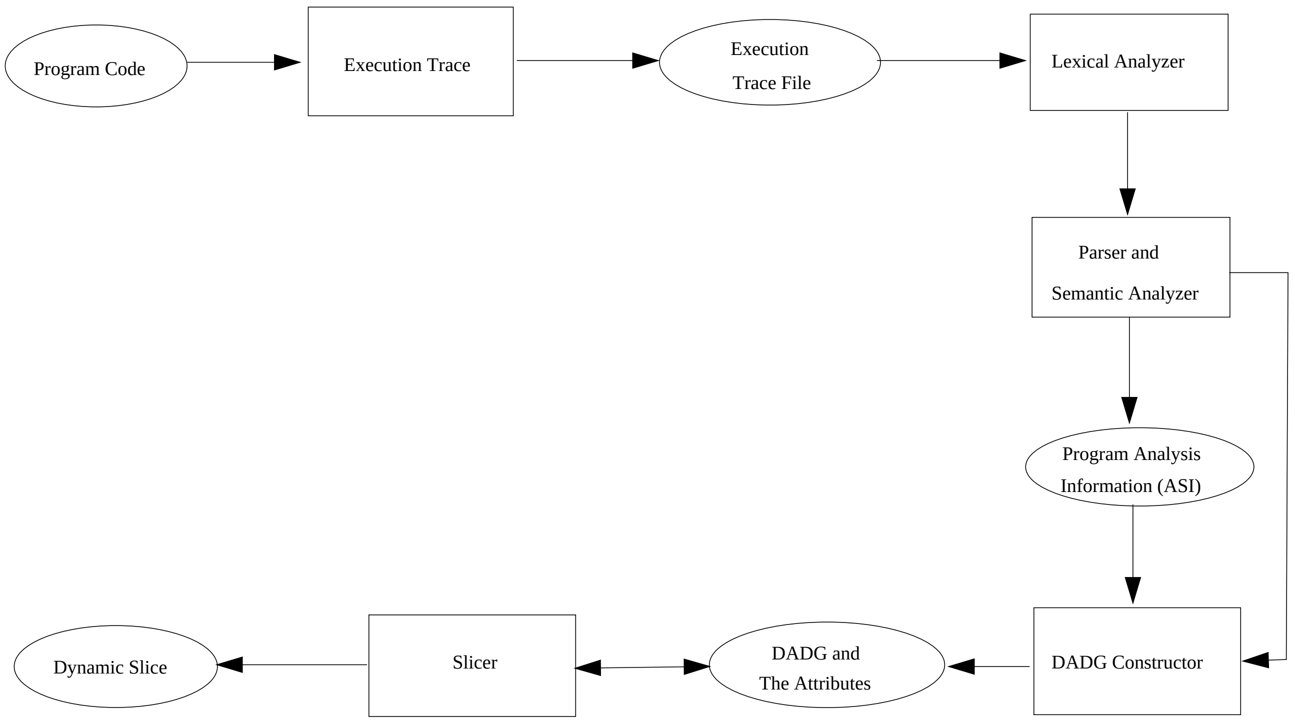


Figure 8: Schematic diagram of the slicing tool

A program written in AspectJ is given as input to DDST. The overall control for the slicer is done through a *coordinator* with the help of a *graphical user interface* (GUI). The *coordinator* takes user input from the GUI, interacts with other relevant components to extract the desired results and returns the output back to the GUI.

The *execution trace* component creates an *execution trace file* for a particular execution of the program. This component takes the user input from the coordinator, stores each executed statement for that input in a file and outputs that file back to the coordinator. This file is called *execution trace file*.

The *lexical analyzer* component reads the execution trace file and breaks it into tokens for the grammar expressed in the parser. When the lexical analyzer component encounters a useful token in the program, it returns the token to the parser describing the type of encountered token.

The *parser and semantic analyzer* component functions as a state machine. The parser takes the token given by the lexical analyzer and examines it using the grammatical rules laid for the input programs. The semantic analyzer component captures the following important information of the program.

- For each vertex u of the program
  - the lexical successor and predecessor vertices of u,
  - the sets of variables defined and used at vertex u,
  - the type of the vertex: assignment or test or method call or return etc.

The lexical component, parser and semantic analyzer component provide the necessary program analysis information to the *DADG constructor* component. The *DADG constructor* component first constructs the CFG and the post-dominator tree of the program using the basic information provided by the lexical and semantic analyzer components. The inter-statement control dependences are captured using the CFG and the post-dominator tree. Then, it constructs the DADG of the program with respect to the trace file along with all the required information to compute slices and stores it in appropriate data structures.

The *slicer* component traverses the DADG. It takes the user input from the *coordinator* and outputs the computed information back to the *coordinator*. The graphical user interface (GUI) functions as a front end to the slicing tool.

7.2 Implementation of the slicing tool

We have implemented our algorithm in Java. We have used the compiler writing tool *ANTLR* (Another Tool for Language Recognition) [1, 32] for *Lexical Analyzer*, *Parser* and *Semantic Analyzer* components of our slicer. ANTLR is a tool that lets one define language grammars in EBNF (Extended Backus-Naur Form) like notations. ANTLR is more than just a grammar definition language. However, the tools provided allow one to implement the ANTLR defined grammar by automatically generating lexers and parsers in Java or other supported languages. ANTLR is a language tool that provides a framework for constructing recognizers, compilers, and translators from grammatical descriptions containing programming languages such as C++, Java, AspectJ etc. ANTLR is a LL(k) based recognition tool.

Table 1: Encoding used for different types of edges of DADG

Code	Edge Type
0	No Edge
1	Control Dependence Edge (True Case)
2	Control Dependence Edge (False Case)
3	Data Dependence Edge (Loop Independent Case)
4	Data Dependence Edge (Loop Carried Case)
5	Weaving Edge

The sample AspectJ program is executed for a given input. The executed statements are stored in a trace file. This trace file is given as input to the ANTLR program. The lexer part of the ANTLR extracts program tokens and stores the data in a data structure called *statement_info*. The DADG of the AspectJ program is automatically constructed by taking input from the *parser* and *semantic analyzer component*. For constructing the DADG, we have used many flags such as *if_flag* to check whether the statement is an if statement or not, *while_flag* to check whether the statement is a while statement or not etc.

We have used an adjacency matrix *dadg[][]* to store the DADG of the given AspectJ program P. This matrix is of the following type:

```
typedef struct edge {
int exist, type;
} edge;
```

- The attribute *exist* has value 0 or 1. `dadg[i][j].exist` is 1 if there is an edge between node number i and j, otherwise 0.
- The data member *type* specifies the type of the edge. The codes used for this are given in Table 1.

We store the following additional information along with the DADG:

- The set *Def(var)* for each variable var in the aspect-oriented program P.
- The set *Use(var)* for each variable var in the aspect-oriented program P.

The sets *Def(var)* and *Use(var)* are stored using arrays.

7.3 Experimental results

With different slicing criteria, the algorithm has been tested on many programs for 40-50 runs. The sample programs contain loops and conditional statements. Table 2 summarizes the *average run-time requirements* of the trace file based algorithm for several programs. Since we have computed the dynamic slices at different statements of a program, we have calculated the average run-time requirements of our trace file based algorithm. The program sizes are small since right now the tool accepts only a subset of AspectJ constructs. However, the results indicate the overall trend of the performance of the trace file based algorithm.

Table 2: Average runtime

Sl No.	Prg. Size (# stmts)	Trace file Based Algorithm (in Sec.)
1	17	0.48
2	43	0.71
3	69	0.92
4	97	1.14
5	123	1.36
6	245	2.46
7	387	3.96
8	562	5.52

The results in Table 2 indicate that the run-time requirement for the trace file based algorithm increases gradually. This is due to the fact that separate vertices are created in the DADG during run-time for different executions of the same statement. This is followed by a depth-first or breadth-first graph traversal on DADG to compute the dynamic slice. Thus, average run-time requirement becomes high since considerable time is required to perform the traversal on DADG. Furthermore, the algorithm uses a trace file to store the execution history. The time required to read the data from a trace file is significant and is added to the average run-time while computing dynamic slice. All these result in the increase of average run-time requirement gradually.

7.4 Comparison with existing work

Very few work has been done on slicing of aspect-oriented programs [22, 23]. Zhao [22] has proposed an intermediate representation called *Aspect-Oriented System Dependence Graph* (ASDG). The ASDG for the example program of Figure 2 as proposed by Zhao [22] is shown in Figure 9. In this ASDG, the *pointcuts* are not represented. But, our DADG correctly represents the pointcuts.

Zhao and Rinard [23] developed an algorithm to construct the SDG for aspect-oriented programs. Figure 10 shows the SDG of the program given in Figure 2 as proposed by Zhao and Rinard [23]. But, the drawback of this

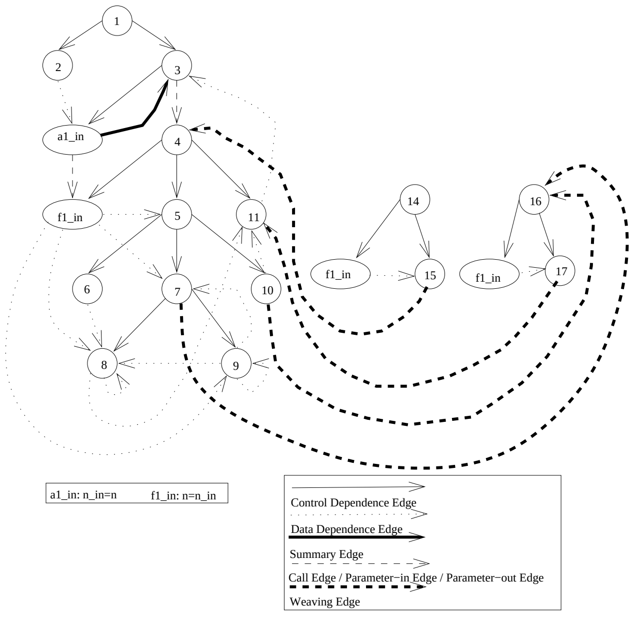


Figure 9: ASDG of the program given in Figure 2 as proposed by Zhao [22]

SDG is that the *weaving process* is not represented correctly. For example, there should be a weaving edge between vertices 15 and 4, beacause, after the execution of *before advice* at statement 14, the actual execution of the method *factorial()* at statement 4 will be started. The use of this SDG to compute dynamic slice results in missing of some statements. So, we cannot use this approach to compute dynamic slice of an aspect-oriented program correctly. Also, they [23] have not considered the dynamic slicing aspects. But in our approach, we have considered the weaving process by adding weaving edges at the appropriate join points in the DADG. Again our algorithm computes precise *dynamic* slices.

8 Conclusion

We proposed an algorithm for dynamic slicing of aspect-oriented programs. First, we have constructed a dependence-based intermediate representation for aspect-oriented programs. We named this representation *Dynamic Aspect-Oriented Dependence Graph* (DADG). Then, we have developed an algorithm to compute dynamic slices of AOPs using the DADG. We have used a trace file to store the execution history. So, we have named our algorithm *Trace file Based Dynamic Slicing* (TBDS) algorithm for AOPs. The resulting dynamic slice in our approach is precise since we create a node in the DADG for each occurrence of a statement in the execution trace. We have developed a tool called *Dynamic Depenedence Slicing Tool* (DDST) to implement our algorithm.

Our algorithm can be extended to compute dynamic

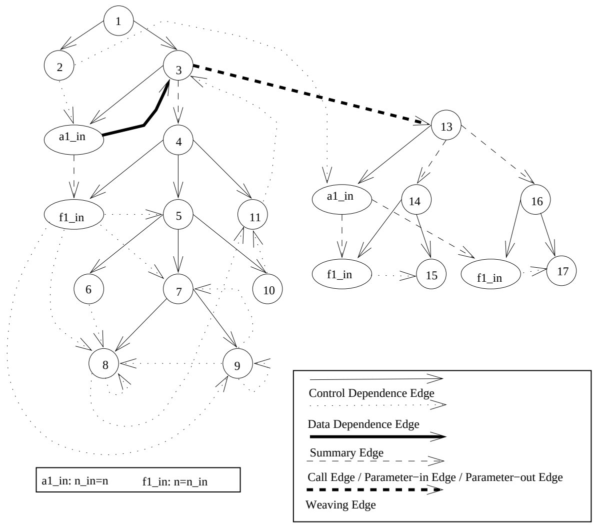


Figure 10: SDG of the program given in Figure 2 as proposed by Zhao and Rinard [23]

slices of concurrent AOPs and distributed AOPs running on different machines connected through a network. The algorithm can also be extended to compute *conditioned slices* with respect to a given condition. Although we have presented the approach for AspectJ, this approach can be easily extended to other aspect-oriented languages such as AspectWerkz, AML, RIDL etc. Our tool can be used to develop efficient debuggers and test drivers for large scale aspect-oriented programs.

References

[1] Antlr. www.antlr.org.

[2] Aspect-Oriented Programming. www.wikipedia.org.

[3] AspectJ. www.eclipse.org/aspectj.

[4] Introduction to AOP. www.media.wiley.com.

[5] Mendhekar A., Kiczales G., and Lamping J. RG: A Case-Study for Aspect-Oriented Programming. Technical report, Xerox Palo Alto Research Center, February 1997.

[6] Dufour B., Goard C., Hendren L., Verbrugge C., Moor O. D., and Sittampalam G. Measuring the Dynamic Behaviour of AspectJ Programs. Technical report, McGill University, School of Computer Science, Sable Research Group and Oxford University, Computing Laboratory, Programming Tools Group, March 2004.

[7] Korel B. and Laski J. Dynamic Program Slicing. *Information Processing Letters*, 29(3):155–163, 1988.

[8] Mund G. B., Mall R., and Sarkar S. Computation of Interprocedural Dynamic Program Slices. *Journal of Information and Software Technology*, 45(8):499–512, June 2003.

[9] Xu B. and Chen Z. Dynamic Slicing Object-Oriented Programs for Debugging. In *Proceedings of 2nd IEEE International Workshop on Source Code Analysis amd Manipulation (SCAM'02)*, pages 115–122, 2002.

[10] Murphy G. C., Walker R. J., and Baniassad E. L. A. Evaluating Emerging Software Development Technologies: Lessons Learned from Assessing Aspect-Oriented Programming. *IEEE Transactions on Software Engineering*, 25(4):438–455, July-Aug 1999.

[11] Liang D. and Harrold M. J. Slicing Objects Using System Dependence Graph. In *Proceedings of the International Conference on Software Maintenance, IEEE*, pages 358–367, November 1998.

[12] Tip F. A Survey of Program Slicing Techniques. *Journal of Programming Languages*, 3(3):121–189, 1995.

[13] Cugola G., Ghezzi C., and Monga M. Coding Different Design Paradigms for Distributed Applications with Aspect-Oriented Programming. In *Proceedings of Workshop on System Distribution: Algorithm, Architecture and Language, WSDAAL*. Italy, 1999.

[14] Kiczales G., Hilsdale E., Hugunin J., Kersten M., Palm J., and Griswold W. G. An Overview of AspectJ. In *Proceedings of the European Conference on Object-Oriented Programming*. Budapest,Hungary, 18-22 June 2001.

[15] Kiczales G., Hilsdale E., Hugunin J., Kersten M., Palm J., and Grisworld W. G. Report on An Overview of Aspectj. Notes by Tai Hu, CMSC631 Fall 2002.

[16] Kiczales G., Irwin J., Lamping J., Loingtier J. M., Lopes C. V., Maeda C., and Mendhekar A. Aspect-Oriented Programming. In *Proceedings of the European Conference on Object-Oriented Programming (ECOOP)*, Finland, June 1997. Springer-Verlag.

[17] Kiczales G. and Mezini M. Aspect-Oriented Programming and Modular Reasoning. In *Proceedings of the 27th International Conference on Software Engineering, ICSE'05*, May 2005.

[18] Agarwal H. and Horgan J. R. Dynamic Program Slicing. In *ACM SIGPLAN Notices, Proceedings of the ACM SIGPLAN 1990 conference on Programming language design and implementation PLDIŠ90*, volume 25 of *6*, pages 246–256, June 1990.

[19] Agrawal H. and Horgan J. R. Dynamic Program Slicing. In *Proceedings of the ACM SIGPLAN'90 Conference on Programming Languages Design and Implementation, SIGPLAN Notices, Analysis and Verification*, volume 25, pages 246–256, 1990.

[20] Irwin J., Loingtier J. M., Gilbert J. R., Kiczales G., Lamping J., Mendhekar A., and Shpeisman T. Aspect-Oriented Programming of Sparse Matrix Code. In *Proceedings of International Scientific Computing in Object-Oriented Parallel Environments (ISCOPE), Marina del Rey, CA*. Springer-Verlag, December 1997.

[21] Zhao J. Dynamic Slicing of Object-Oriented Programs. Technical report, Information Processing Society of Japan, May 1998.

[22] Zhao J. Slicing Aspect-Oriented Software. In *Proceedings of 10th International Workshop on Program Comprehension*, pages 251–260, June 2002.

[23] Zhao J. and Rinard M. System Dependence Graph Construction for Aspect-Oriented Programs. Technical report, Laboratory for Computer Science, Massachusetts Institute of Technology, USA, March 2003.

[24] Larsen L. and Harrold M. J. Slicing Object-Oriented Software. In *Proceedings of 18th International Conference on Software Engineering*, pages 495–505, March 1996.

[25] Sahu M. and Mohapatra D. P. A Node Marking Technique for Dynamic Slicing of Aspect-Oriented Programs. In *Proceedings of the 10th International Conference on Information Technology (ICIT'07)*, pages 155–160, 2007.

[26] Weiser M. Programmers Use Slices When Debugging. *Communications of the ACM*, 25(7):446–452, July 1982.

[27] Mohapatra D. P. *Dynamic Slicing of Object-Oriented Programs*. PhD thesis, Indian Institute of Technology, Kharagpur, May 2005.

[28] Mohapatra D. P., Mall R., and Kumar R. Dynamic Slicing of Concurrent Object-Oriented Programs. In *Proceedings of International Conference on Information Technology: Progresses and Challenges (ITPC)*, pages 283–290. Kathamandu, May 2003.

[29] Mohapatra D. P., Mall R., and Kumar R. An Edge Marking Technique for Dynamic Slicing of Object-Oriented Programs. In *Proceedings of the 28th Annual International Computer Software and Applications Conference (COMPSAC'04)*, 2004.

[30] Mohapatra D. P., Mall R., and Kumar R. A Node-Marking Technique for Dynamic Slicing of Object-Oriented Programs. In *Proceedings of Conference on Software Design and Architecture (SODA'04)*, 2004.

[31] Mohapatra D. P., Mall R., and Kumar R. An Overview of Slicing Techniques for Object-Oriented Programs. *Informatica*, 30:253–277, 2006.

[32] Levine J. R. Mason T. and Brown D. *Lex and Yacc*. O'REILLY, 3rd edition, 2002.

[33] Horwitz S., Reps T., and Binkley D. Inter-Procedural Slicing Using Dependence Graphs. *ACM Transactions on Programming Languages and Systems*, 12(1):26–60, January 1990.

[34] Braak T. T. Extending Program Slicing in Aspect-Oriented Programming with Inter-Type Declarations. 5th TSConIT Program, June 2006.

[35] Ishio T., Kusumoto S., and Inoue K. Program Slicing Tool for Effective Software Evolution Using Aspect-Oriented Technique. In *Proceedings of Sixth International Workshop on Principles of Software Evolution*, pages 3–12. IEEE Press, September 2003.

[36] Song Y. T. and Huynh D. T. Forward Dynamic Object-Oriented Program Slicing. In *Proceedings of IEEE Symposium on Application Specific Systems and Software Engineering and Technology (ASSET'99)*, pages 230–237, Richardson, TX, USA, 03/24/1999-03/27/1999 1999.

[37] Wang T. and Roychoudhury A. Using Compressed Bytecode Traces for Slicing Java Programs. In *Proceedings of 26th International Conference on Software Engineering (ICSE'04)*, pages 512–521, 23-28 May 2004.

[38] Binkley D. W. and Gallagher K. B. Program Slicing. *Advances in Computers*, 43, 1996. Academic Press, San Diego, CA.

[39] Chen Z. and Xu B. Slicing Object-Oriented Java Programs. *ACM SIGPLAN Notices*, 36(4), April 2001.

Recognition of On-line Handwritten Arabic Digits Using Structural Features and Transition Network

Al-Taani Ahmad
Department of Computer Sciences, Yarmouk University, Jordan
E-mail: ahmadta@yu.edu.jo

Hammad Maen
Department of Computer Sciences, the Hashemite University, Jordan
E-mail: maen@hu.edu.jo

Keywords: on-line digit recognition, pattern recognition, feature extraction, structural primitives, document processing, transition networks

Received: September 12, 2006

In this paper, an efficient structural approach for recognizing on-line handwritten digits is proposed. After reading the digit from the user, the coordinates (x, y) of the pixels representing the drawn digit are used for calculating and normalizing slope values of these coordinates. Successive slope values are then used to record the change of direction which used to estimate the slope. Based on the changing of signs of the slope values, the primitives are identified and extracted. These primitives represent a specific string which is a production of a certain grammar. Each digit can be described by a specific string. In order to identify the digit we have to determine to which grammar the string belongs. A Finite Transition Network which contains the grammars of the digits is used to match the primitives' string with the corresponding digit to identify the digit. Finally, if there is any ambiguity, it will be resolved. The proposed method is tested on a sample of 3000 digits written by 100 different persons; each person wrote the 10 digits three times each. The method achieved accuracy of about 95% on the sample test. Experiments showed that this technique is flexible and can achieve high recognition accuracy for the shapes of the digits represented in this work.

Povzetek: V prispevku je opisana metoda prepoznavanja arabskih črk.

1 Introduction

In areas of automatic document analysis and recognition, the correct interpretation of digits is very important. Automatic recognition of on-line handwriting has a variety of applications at the interface between man and machine.

The performance of any system for handwriting recognition can be evaluated by several factors, such as size of the alphabet, independence of the writing style, and speed of recognition.

Automatic recognition of handwritten digits is difficult due to several reasons, including different writing styles of different persons, different writing devices, and the context of the digit. This leads to digits of different sizes and skews, and strokes that vary in width and shape.

Researchers in this field have proposed different approaches, such as statistical, structural, and neural network approaches [1, 2]. The main primitives that form digits are line segments and curves. Different arrangements of these primitives form different digits. To recognize a digit, we should first determine the structural relationships between the features make up the digit.

The syntactic and structural approaches require efficient extraction of primitives [3-5].

In this study, we propose an efficient approach for extracting features for handwritten digits recognition. First, we will review some related work. After introducing the Normalization and slope estimation method used in this paper, we will discuss the feature extraction algorithm used to extract the primary and secondary features. Then, we will give an overview of the proposed recognition approach. We will also illustrate how to resolve ambiguities in some digits. Finally, we will present and discuss the experimental results and draw some conclusions.

2 Previous works

The problem of handwriting recognition has been studied for decades and many methods have been developed. Verma [6] proposed a contour code feature in conjunction with a rule based segmentation for cursive handwriting recognition. A heuristic segmentation algorithm is used to segment each word. Then the segmentation points are passed through the rule-based module to discard the incorrect segmentation points and include any missing segmentation points.

You et al. [7] presented an approach for segmentation of handwritten touching numeral strings. They designed a neural network to deal with various types of touching observed frequently in numeral strings.

A numeral string image is split into a number of line segments while stroke extraction is being performed and the segments are represented with straight lines. Segmentation points are located using the neural network by interpreting the features collected from the primitives.

Olszewski [8] proposed a recognition approach that uses syntactic grammars to discriminate among digits for extracting shape-based or structural features and performing classification without relying on domain knowledge. This system employs a statistical classification technique to perform discrimination based on structural features is a natural solution. A set of shape-based features is suggested as the foundation for the development of a suite of structure detectors to perform generalized feature extraction for pattern recognition in time-series data.

Chan et al. [9] proposed a syntactic approach to structural analysis of on-line handwritten mathematical expressions. The authors used definite clause grammar (DCG) to define a set of replacement rules for parsing mathematical expressions. They also proposed some methods to increase the efficiency of the parsing process. The authors tested the proposed system on some commonly seen mathematical expressions and they claimed that their method has achieved satisfactory results, making mathematical expression recognition more flexible for real-world applications.

Chan et al. [10] discussed a structural approach for recognizing on-line handwriting. The recognition process starts when getting a sequence of points from the user and then by using these points to extract the structural primitives. These primitives include different types of line segments and curves. The authors demonstrated their approach on 62 character classes (digits, uppercase and lowercase letters). Each class has 150 different entries. They stated that experimental results showed that the recognition rates were 98.60% for digits, 98.49% for uppercase letters, 97.44% for lowercase letters, and 97.40% for the combined set.

Amin [11] reviewed the state of Arabic character recognition research throughout the last two decades. The author summarized all the work accomplished in the past two decades in off-line systems in an attempt to pin-out the different areas that need to be tackled.

Behnke et al. [12, 13] proposed a case study on the combination of classifiers for the recognition of handwritten digits. Four different classifiers are used and evaluated; Wavelet-Preprocessing Classifier, Structural Classifier, Neural Networks Classifier, and Combined Classifier.

3 Overview of the proposed system

3.1 Normalization and slope estimation

The user draws the digit on a special window using a digitizer (or mouse). Then the coordinates (x, y) of the pixels representing the drawn digit are saved on a file. These coordinate values are used for calculating and normalizing slope values. Successive slope values are then used to record the change of direction which used to estimate the slope [14].

The signs of the slope values (+ and -), the zero values, and the infinity values are saved and used in the feature extraction step. Figure 1 shows an example, the representation of the digit 2. Then, all primitives representing each digit are extracted. These primitives are identified by locating break points in the digit. Two types of break points are identified: Primary Break Points (PBP): slope values of infinity (∞) and Secondary Break Points (SBP): slope values of zero. The feature extraction process depends on the change of the slope signs around these break points. Infinity breakpoint is considered to be primary breakpoint since all primary features (Figure 3) needed for recognition have a slope value of infinity, and all the secondary features (Figure 5) have a slope of zero.

After the slope signs and values are computed, these values are normalized. The purpose of this step is to eliminate any distortion that might occur during the drawing process and to ease the primitive identification

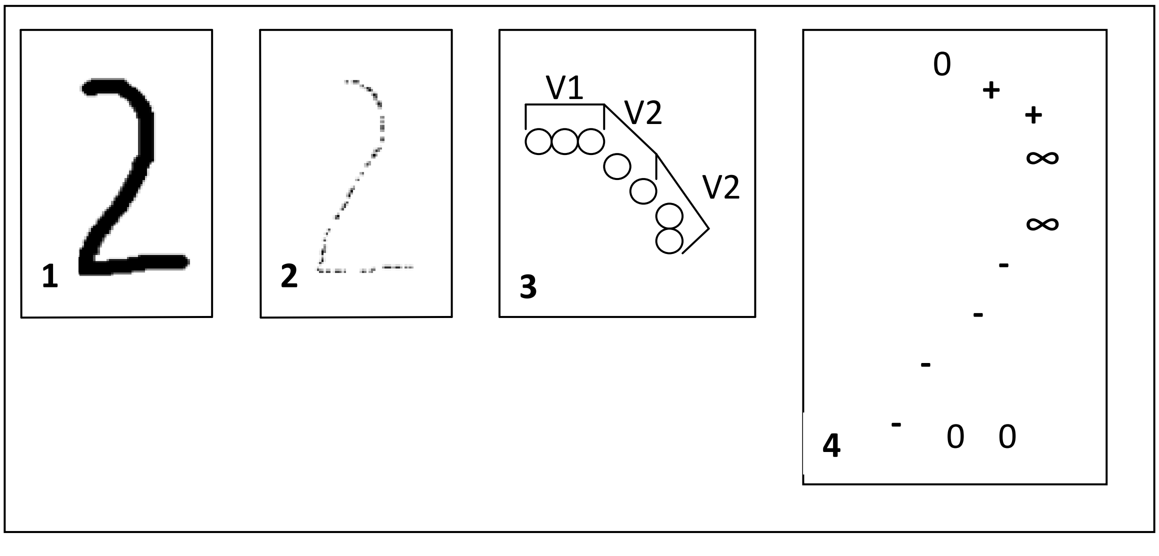


Figure 1: Representation of the digit 2

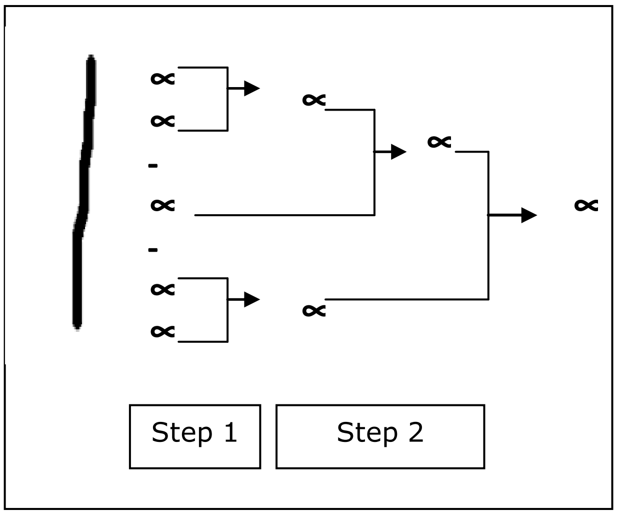


Figure 2: Removing Redundant Break Points

process and guarantee accurate identification. Two steps of normalization are done:

1. Removing redundant break points: Eliminating adjacent reference points of the same type, except the first one. This step is required to record the change of the signs; only one break point is needed.
2. A threshold value is used to determine the distance (number of slope value signs) between any two successive break points of the same type.

Figure 2 shows an example of these two steps for the digit 1. In step 1, the adjacent break points are removed, then in step 2 the threshold value was used to remove more redundant break points since the distance is less than the threshold value, e.g. the slope values between -0.1 and +0.1 is saved as zero. The result is only one break point with two different slope sings before and after it. This result will be used in the primitive identification process.

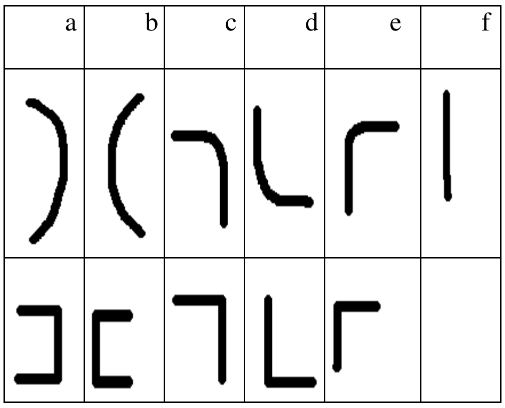


Figure 3: Primary Primitives

Final representation

In addition to the signs of the slope values and break points, the X and the Y positions for the middle pixel in which its neighbors (from both sides) used to calculate the slope. So the slope value is saved with its X and Y positions. Also, the sign of ΔY i.e. (Y2 – Y1) is used to determine the direction of writing or drawing (upward or downward). If $\Delta Y > 0$ and the reference point (0, 0) is located on the top left corner, then the directing of writing is downward, otherwise the directing of writing is upward. The final representation of the digit is represented as a list of vectors V1, V2, ... , Vn; each vector V contains the following data: (slope value (sign or break point), Y position, X position, ΔY sign) [15].

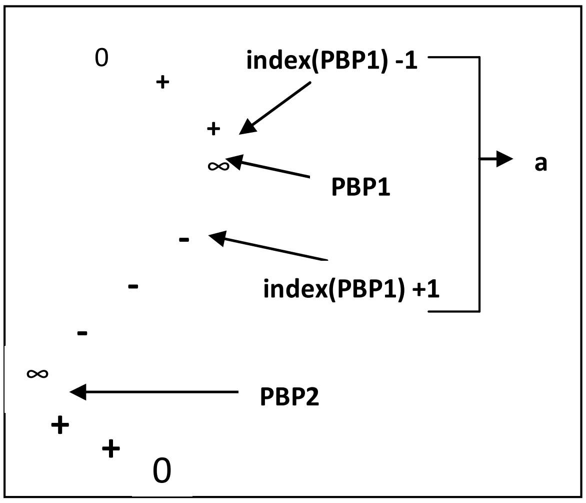


Figure 4: Signs and Break Points for the digit 2

3.2 Extracting primary primitives

The final representation of the digit is used to extract primary primitives [16]. Figure 3 shows these primitives (a, b, c, d, e, or f). These primitives are called primary primitives because the primary break points (PBP) are used to identify them.

To extract the primitives we use the following algorithm:

```
For each PBP do:
  IF the slope sign before it is (+) and after it is (-)
    then the primitive is 'a'.
  Else if the slope sign before it is (-) and after it is (+)
    then the primitive is 'b'
  Else if the successive slope signs before it is (+) and
    end with SPB then the primitive is d'.
  Else if the successive slope signs after it is (-) and
    end with SPB then the primitive is 'e'.
  Else if the successive slope signs after it is (+) and
    end with SPB then the primitive is 'c'.
```

Figure 4 explains this step for the digit 2. Assume that

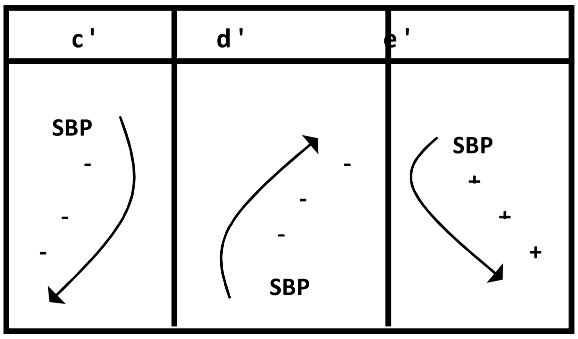


Figure 6: Identification process

the user draw the digit downward. In this case, ΔY is greater than zero for all points, so the algorithm proceeds as follows:

1. Take the first PBP1.
2. Now, primitive "a" is recognized.
3. Find the next PBP.
4. Take the next PBP (PBP2).
5. Now, primitive "b" is recognized.
6. No More PBP, end.

Now the vector contains the primitives "ab". If the user draws the digit from bottom to top, the vector will contain the primitives "ba". We need also to extract starting and ending points for the actual curves drawn that represented by primitives "a" and "b". This step is necessary to resolve ambiguity which will be explained in section 3.6. Each digit has different patterns which captured by the set of primitives that are described in Figure 3.

After feature extraction process, we need to identify these features. The primitives which are extracted in the previous phases represent a certain string which is a production of a certain grammar.

3.3 Extracting secondary primitives

After the process of extracting primary primitives finishes, another extracting process begins to identify another category of primitives called Secondary Primitives. Secondary Break Points (SBP), the zero values, and the slope signs around them are used to identify these primitives which shown in Figure 5. For example, the Secondary Primitive "c' " is the same as the primary primitive "c" but here there is no PBP and the primitive makes acute angle with SBP.

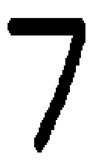
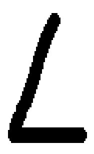
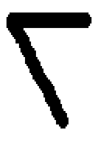
c'	d'	e'
		

Figure 5: Secondary Primitives

To extract these primitives we use the following algorithm (Figure 6):

```
FOR each SBP that is not part of any primary primitive:
    IF the slope signs after it are (-) and make lower
        acute angle then the primitive is "c'".
    Else, if the slope signs after it are (+) and make lower
        acute angle then the primitive is "e"'.
    Else, if the slope signs before it are (+) and make
        upper acute angle then the primitive is "d'".
```

3.4 Sorting primitives

At this stage, the primitives' vector contains primary and secondary primitives. The order of these primitives depends on the drawing style. For example, if the drawing style was downward (when drawing digit 2) then the primitives' vector will contain "ab", on the other hand if the drawing style was upward, it would contain "ba". This is confusing and increases the number of patterns for the digit. The order is very important in translating the primitives into digits. So we need a standard order to be used in sorting all primitives.

The order of the identified primitives must be independent from the drawing style and from the order of drawing the primitives. The standard order used here is the Y position for the break points; they are sorted in increasing order. After collecting all primitives, they are reordered according to the Y position for the break points that used to identify them. When two break points have the same Y position the order is based on the order of drawing. Indeed this is not a problem, because each character has many patterns, as we will see, to deal with such problems.

3.5 Identifying the digit

Each digit has different patterns which captured by the set of primitives, these patterns are shown in Figure 7. These primitives represent a specific string which is a production of a certain grammar. Each digit can be described by a specific string. In order to identify the digit we have to determine to which grammar the string belongs. A transition network has been used to match the primitives' string with corresponding digit. This network is shown in Figure 8.

3.6 Distinguish ambiguous digits

As we can see in Figure 8, there are multiple digits which have the same string of primitives, for example the string "ab" is common for digits 0 and 2. In this phase this ambiguity is removed, and more constrains on some digits are applied to guarantee the correct result. The key elements, that help us in resolving this ambiguity, are the starting and ending points, x-y coordinates, of the actual curves representing primitives "a" and "b". This process is done in phase (section) 3.2. Figures 9 and 10 show these points.

3.6.1 Ambiguity in "ab"

From Figure 8 we can see that the digits 0 and 2 both have one string "ba" which is one of their shapes. Now, the question is how this ambiguity can be distinguished? Assume that, as in Figure 9:

- (a1) is the x-y coordinates for the starting point of curve a.
- (a2) is the x-y coordinates for the ending point of curve a
- (b1) is the x-y coordinates for the starting point of curve b
- (b2) is the x-y coordinates for the ending point of curve b.

By using these definitions, we can easily distinguish between 2 and 0. Figure 10 shows the distinguishing criteria between digits 2 and 0.

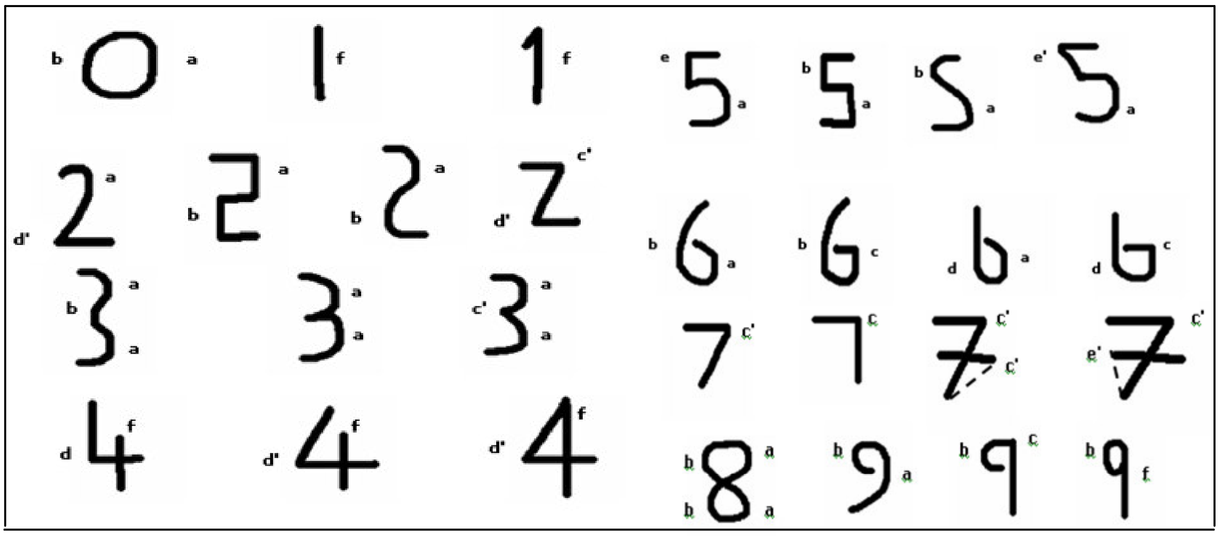


Figure 7: Possible handwritten patterns

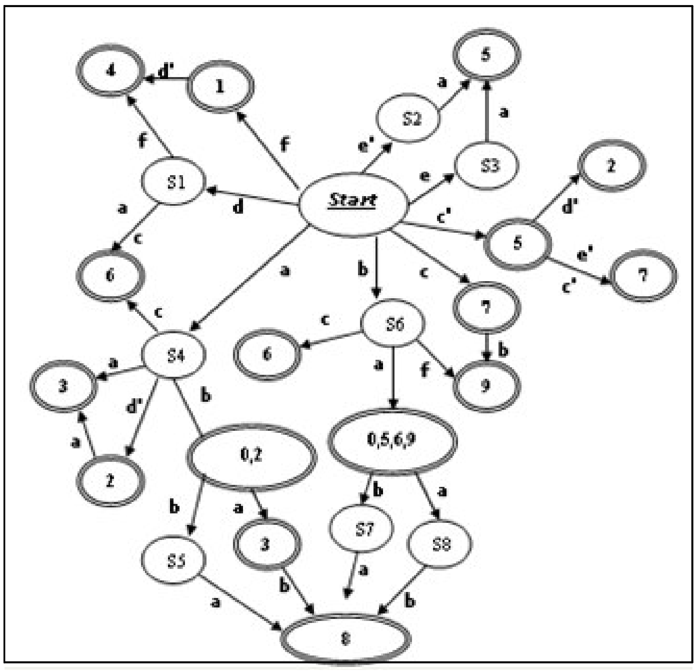


Figure 8: Transition Network

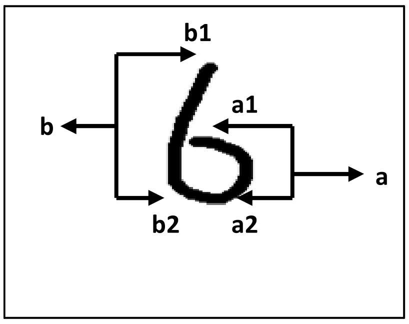


Figure 9: Ending Points

3.6.2 Ambiguity in "ba"

As we can see in Figure 8 the production string “ba" gives us the digits 0, 5, 6, and 9. The same definitions described in the above, in “ab”, are used here. The process is more complicated since we have 4 digits. The distinguishing criteria are described in Figure 11. The digit 2 can be easily distinguished by checking the point b1 if it above a1 or not. Now, to distinguish the three remaining digits we use the distance between the ending points of the drawing curves.

4 Experimental results and discussions

We used a digitizer with special software with a 1.6 MHZ PC. One hundred different persons tested the program. Each one of them drew all the numbers 3 times. So, for each number the program attempts to recognize it 300 times. As a result, there is only one possible outcome in the recognition process: correct or incorrect. We summarized the testing result for all digits in Figure 12. We can see that the system ability to recognize the drawing shape, shown in Figure 7, correctly is about 95%. There are 5% incorrect results, these results include both results; cannot identifying the drawing or identifying it incorrectly.

From the testing process we noticed the following important remarks:

1. The drawing speed may affect on the recognition process. If the user draws very quickly, the system might not capture all the input pixels representing the digit, i.e. the drawing must be connected, so the user has to draw the digit as one connected line. Only the digits 4 and 7 can be disconnected because

we included the suitable patterns to deal with the disconnected line in 4 and 7.

2. The accuracy rate for digits 6 and 9 is quite low. This is because they have the same patterns, i.e. they have the same production string "ba" (see Figure 11).
3. The proposed system works only on Arabic digits. We did not consider non-digits like letters and special characters. This task is left for future work

It can be noticed that the accuracy of the proposed approach is lower than the accuracy of Chan et al. [10] work in which they have achieved a recognition rate of 98.60%. The accuracy of the system depends on many factors like whether there is noise in the test data, if the digit is poorly written, deliberately written in some strange and unusual way, or with zig-zag line segments. We should take also into account that the writing process itself is subjective and depends on the person writing style. If the test data are carefully selected then the system could give higher accuracy rate.

Despite these factors our approach has the following advantages:

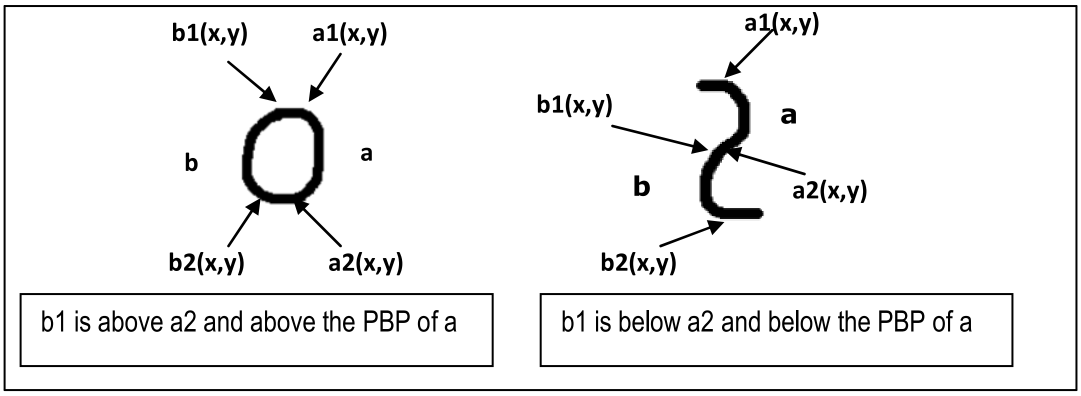

b1(x,y)
a1(x,y)
b
a
b2(x,y)
a2(x,y)
b1 is above a2 and above the PBP of a
a1(x,y)
b1(x,y)
a
b
a2(x,y)
b2(x,y)
b1 is below a2 and below the PBP of a

Figure 10: Resolving ambiguity in "ab"

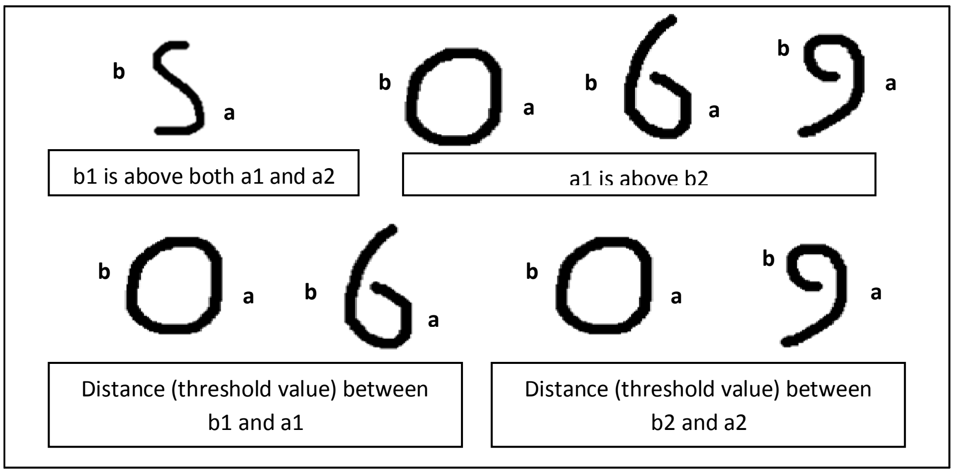

b
a
b1 is above both a1 and a2
b
a
b
a
b
a
a1 is above b2
b
a
b
a
Distance (threshold value) between b1 and a1
b
a
b
a
Distance (threshold value) between b2 and a2

Figure 11: Resolving ambiguity in "ba"

Digit	Correct	Percent
1	288/300	0.96
2	290/300	0.97
3	289/300	0.96
4	288/300	0.96
5	285/300	0.95
6	274/300	0.91
7	287/300	0.96

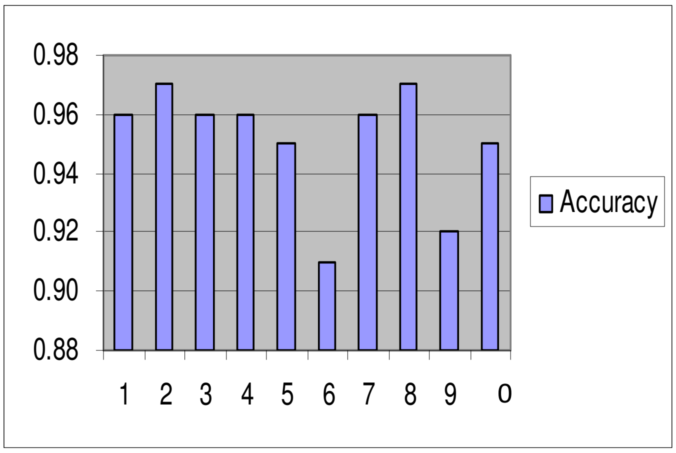


Figure 12: Test Results

1. In the proposed approach, we used shape-based features like curve, line, dot ... etc. together with a flexible Transition Network Grammar in the recognition process. Our experiment demonstrated that the use of shape-based features achieve fairly good recognition results since these features are used by people visually to recognize digits. Also, we used Transition Network since it works in the same manner as the human information processing system does which reflects one of our main objectives in this work, to design an intelligent agent which behaves rationally like humans.
2. The proposed approach can be modified to work on letters and other characters.
3. The proposed approach is unsupervised, i.e. training is not necessary.

5 Conclusions and future work

A new online structural pattern recognition approach is discussed. This approach recognizes the handwritten digits; the primitives are determined by identifying the changes in the slope's signs around the zero and the infinity values (break points). This technique is independent of the type of drawing (upward or downward). A special grammar has been used to match the string of primitives to the corresponding digit. The method is tested on an on-line dataset representing the digits 0-9 collected from 100 users. On the average, the recognition rate was about 95%. Future work considers testing the method on a larger data set to improve the effectiveness of the method, since we get most of the writing variations of the digits by different users.

The proposed method will be modified to deal with Arabic handwritten characters. In addition, the next important work is to add additional constraints on the primitives, for example the average length of one primitive according to another and do the primitives connected correctly or not. These constrains can guarantee an accurate results and do not directly match the resulting string of primitives to its corresponding digit unless the primitives form the digit correctly, one important note here is that, more constraints may reduce the probability of recognition

References

[1] C. C. Tappert, C. Y. Suen, and T. Wakahara. The state of the art in on-line handwriting recognition. *IEEE Trans. On Pattern Analysis and Machine Intelligence*, 12 (8), pp. 787-808, 1990.

[2] R. G. Casey and E. Lecolinet. Strategies in character segmentation: A survey. *Proceedings of International Conference on Document Analysis and Recognition*, pp. 1028-1033, 1995.

[3] K. S. Fu. *Syntactic Pattern Recognition and Applications*. Prentice-Hall, Englewood Cliffs, NJ, 1982.

[4] T. Pavlidis. *Structural Pattern Recognition*. Springer, New York, 1977.

[5] S. Lucas, E. Vidal, A. Amiri, S. Hanlon, and J.C. Amengual. A comparison of syntactic and statistical techniqes for off-line OCR. in: R. C. Carrasco, J. Oncina (Eds.), *Grammatical Inferrence and Applications (ICGI-94)*, Springer, Berlin, pp. 168-179, 1994.

[6] Brijesh Verma. A Contour Code Feature Based Segmentation For Handwriting Recognition. *Proceedings of the Seventh International Conference on Document Analysis and Recognition (ICDAR 2003)*, Vol. 1, PP. 1203 – 1207. 2003.

[7] Daekeun You and Gyeonghwan Kim. An approach for locating segmentation points of handwritten digit strings using a neural network. *Proceedings of the Seventh International Conference on Document Analysis and Recognition (ICDAR 2003),* Vol. 1, PP. 142 – 146.

[8] Robert T. Olszewski. Generalized Feature Extraction for Structural Pattern Recognition in TimeSeries Data. *PhD thesis*, University-Pittsburgh, 2001.

[9] Kam-Fai Chan and Dit-Yan Yeung. An effecient syntactic approach to structural analysis of on-line handwritten mathematical expressions. *Pattern Recognition,* Vol. 33, pp. 375 - 384, 2000.

[10] Kam-Fai Chan and Dit-Yan Yeung. Recognizing on-line handwritten alphanumeric characters through flexible structural matching. *Pattern Recognition,* Vol 32, pp. 1099 - 1114, 1999.

[11] Adnan Amin. Off-Line Arabic Character Recognition: The State Of The Art. *Pattern Recognition*, 31 (5), pp. 517 - 530, 1998.

[12] Sven Behnke, Marcus Pfisher, and Raul Rojas, "A Study on the Combination of Classifiers for Handwritten Didit Recognition", Proceedings of Neural Networks in Applications, Third International Workshop (NN'98), Magdeburg, Germany, pp. 39-46, 1998.

[13] Sven Behnke, Raul Rojas, and Marcus Pfister. Recognition of Handwritten Digits using Structural Information. *Proceedings of the International Conference of Neural Network, Houston TX*, Vol. 3, pp. 139 1- 1396, 1997.

[14] S. Madhvanath, G. Kim, and V. Govindaraju. Chaincode Contour Processing for Handwritten Word Recognition. *IEEE Trans. On Pattern Analysis and Machine Intelligence*, 21 (9), pp. 928 - 932, 1999.

[15] Robert Schalkoff. *Pattern Recognition: Statistical, Structural, and Neural Approaches.* John Wiley and Sons Inc. 1992.

[16] Rafael C. Gonzalez and Michael G. Thomason. *Syntactic Pattern Recognition: An Introduction.* Addison Wesley Publishing Company, 1978.

[17] H. Freeman. Computer processing of line drawing images. *ACM Computing Surveys (CSUR),* 6 (1), pp. 57 – 97, 1974.

Robust Speech Recognition Using Perceptual Wavelet Denoising and Mel-frequency Product Spectrum Cepstral Coefficient Features

Mohamed Cherif Amara Korba
University of Larbi Tebessi, Tebessa, Electronic Department
Constantine road, BP 15454, Tebessa, Algeria
E-mail: Amara_korba_cherif@yahoo.fr

Djemil Messadeg
University of Badji-Mokhtar, Annaba, Electronic Department
BP 12, 23000, Annaba, Algeria
E-mail: messadeg@yahoo.fr

Rafik Djemili
University 20 Aout 1955, Skikda, Electronic Department
El-Hadaiek road, BP 26 Skikda, Algeria
E-mail: djemili_rafik@yahoo.fr

Hocine Bourouba
University of Mentouri, Constantine, Electronic Department
Ain El Bey road, BP 325, 25017 Constantine, Algeria
E-mail: bourouba2004@yahoo.fr

Keywords: noise robust speech parametrization, perceptual wavelet-packet transform, penalized threshold, mel-frequency product spectrum cepstral coefficients

Received: May 25, 2008

To improve the performance of Automatic Speech Recognition (ASR) Systems, a new method is proposed to extract features capable of operating at a very low signal-to-noise ratio (SNR). The basic idea introduced in this article is to enhance speech quality as the first stage for Mel-cepstra based recognition systems, since it is well-known that cepstral coefficients provided better performance in clean environment. In this speech enhancement stage, the noise robustness is improved by the perceptual wavelet packet (PWP) based denoising algorithm with both type of thresholding procedure, soft and modified soft thresholding procedure. A penalized threshold was selected. The next stage of the proposed method is extract feature, it is performed by the use of Mel-frequency product spectrum cepstral coefficients (MFPSCCs) introduced by D. Zhu and K.K and Paliwal in [2]. The Hidden Markov Model Toolkit (HTK) was used throughout our experiments, which were conducted for various noise types provided by noisex-92 database at different SNRs. Comparison of the proposed approach with the MFCC-based conventional (baseline) feature extraction method shows that the proposed method improves recognition accuracy rate by 44.71 %, with an average value of 14.80 % computed on 7 SNR level for white Gaussian noise conditions.

Povzetek: Opisana je nova metoda robustnega strojnega prepoznavanja govora.

1 Introduction

ASR systems are used in many man–machine communication dialog applications, such as cellular telephones, speech driven applications in modern offices or security systems. They give acceptable recognition accuracy for clean speech, their performance degrades when they are subjected to noise present in practical environments [3].

Recently many approaches have been developed to address the problem of robust speech parametrization in ASR, The Mel-frequency cepstral coefficients (MFCCs) are the most widely used features, they were adopted in many popular speech recognition systems by many researchers, such as [8],[9]. However, it is well-known that MFCC is not robust enough in noisy environments, which suggests that the MFCC still has insufficient sound representation capability, especially at low SNR.

MFCCs are derived from the power spectrum of the speech signal, while the phase spectrum is ignored. This is done mainly due to our traditional belief that the human auditory system is phase-deaf, i.e., it ignores phase spectrum and uses only magnitude spectrum for speech perception [1]. Recently, it has been shown that the phase spectrum is useful in human speech perception [2]. The features derived from either the power spectrum or the phase spectrum have the limitation in representation of the signal.

In this paper, we proposed noise robust feature extraction algorithm based on enhancement speech signal before extraction feature to improve performance of Mel-cepstra based recognition systems.

The feature extraction system performs two major functions. The first is speech enhancement; the other is feature extraction. (see Figure 1).

The speech enhancement stage employs the perceptual wavelet packet transform (PWPT) instead of conventional wavelet-packet transform, to decompose the input speech signal into critical sub-band signals. Such a PWPT is designed to match the psychoacoustic model and to improve the performance of speech denoising [11]. Denoising is performed by thresholding algorithm introduced by Donoho [7] as a powerful tool in denoising signals degraded by additive white noise.

Denoising procedure is divided into two steps: firstly, threshold is estimated by penalized threshold algorithm [5], and secondly, two types of thresholding algorithms are applied, soft thresholding algorithm [6] and modified soft thresholding (Mst) algorithm proposed in [4] to determine who of these algorithm is more efficient to improve recognition accuracy. Finally, these thresholded wavelet coefficients are constructed to obtain the enhanced speech samples by the inverse perceptual wavelet packet transform (IPWPT).

Stage of feature extraction is performed by the use of Mel-frequency product spectrum cepstral coefficients (MFPSCCs) introduced by D. Zhu and K.K. Paliwal in [2]. This is defined as the product of the power spectrumand the group delay function (GDF). It combines the magnitude spectrum and the phase spectrum.

The GDF can be defined as follows [2]

$$\tau_p(\omega) = -\operatorname{Im}\frac{d(\log(X(\omega))}{d\omega} \tag{1}$$

$$= -\operatorname{Im}\frac{X_R(\omega)Y_R(\omega)+X_I(\omega)Y_I(\omega)}{|X(\omega)|^2} \tag{2}$$

Where *X(ω)* is the Fourier transforms of frame speech *x(n)*, *Y(ω)* is the Fourier transforms of *nx(n)*, and the subscripts *R* and *I* denote the real and imaginary parts.

They have shown in their experiments [2] that the MFPSCC feature gives better performance than power spectrum and phase spectrum features. But in the low SNR the recognition accuracy rate remains weak.

The rest of this paper is organized as follows. Section 2 introduces a block diagram of proposed noise robust feature (PNRF) extraction algorithm and provides detailed description of each constituting part. Section 3 shows a graphical comparison between different features. Section 4 evaluates the performance of the proposed system under a different level of noise conditions. The conclusion is presented in Section 5.

2 Description of proposed feature extraction algorithm

Figure1 presents a block diagram of proposed noise robust feature extraction algorithm. Noisy input speech is sampled at $F_s = 11025\,\text{Hz}$ and segment into frames of length L = 275 samples (25 ms) with frame shift interval of S = 110 samples (10 ms). There is no need to apply classical windowing operation in the perceptual wavelet packet decomposition (PWPD) scheme.

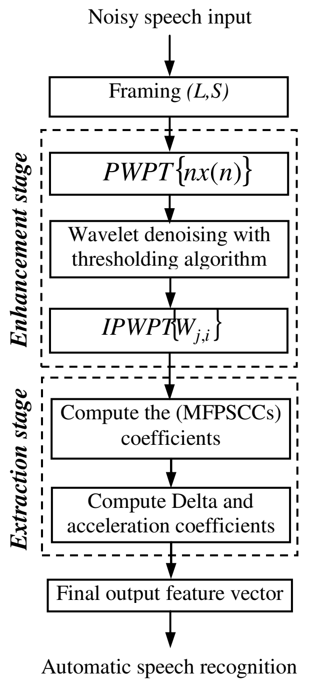

Noisy speech input
Framing (L,S)
Enhancement stage
PWPT{nx(n)}
Wavelet denoising with thresholding algorithm
IPWPT{W_{j,i}}
Extraction stage
Compute the (MFPSCCs) coefficients
Compute Delta and acceleration coefficients
Final output feature vector
Automatic speech recognition

Figure 1: Block diagram of proposed noise robust feature extraction algorithm.

2.1 Perceptual wavelet-packet transform

The decomposition tree structure of PWPT is designed to approximate the critical bands (CB) as close as possible in order to efficiently match the psychoacoustic model [12] [13]. Hence, the size of PWPT decomposition tree is directly related to the number of critical Bands. The sampling rate is 11025 Hz, yielding a speech bandwidth of 5.512 KHz. Within this bandwith, there are approximately 17 critical bands , which are derived from the Daubechies wavelet 8 (db8) and the decomposition is implemented by an efficient 5 level tree structure, the corresponding PWPT decomposition tree can be constructed as depicted in Figure 2. The perceptual wavelet transform (PWT) is used to decompose *nx(n)* into several frequency subbands that approximate the critical bands. The set of wavelet expansion coefficients is generated from

$$\{w_{j,i}(k)\} = PWPT(nx(n)) \tag{3}$$

Where $n = 1, 2, \ldots, L$ (L is the length of frame as mentioned above $L = 275$ samples).
$j = 0, 1, 2, \ldots, 5$ (j: number of levels (five levels)).
$i = 1, 2, \ldots, (2^j - 1)$ (i: denotes index of subbands in each level j).
Terminal nodes of PWPD tree represent a non uniform filterbank, which is sometimes called as 'perceptual filterbank' in the literature. Node (5,0) through (3,7) at the last level of decomposition tree are the terminal node. The output of this stage is a set of wavelet coefficients.

2.2 Wavelet denoising procedure

Denoising by wavelet is performed by thresholding algorithm, in which coefficients smaller than a specific value, or threshold, will be shrunk or scaled [6] ,[14]. There are many algorithms for obtaining threshold value. In this study, threshold is obtained by PWP coefficients using a penalization method provided by Birge-Massart [5].

2.2.1 Penalized threshold for PWP denoising

Let column vector $w_{j,i}$ be a wavelet packet coefficient (WPC) sequence, where j represents wavelet packet decomposition (WPD) level and i stands for sub band.
The standard deviation σ is estimated in the same way as in [6]

$$\sigma = \frac{1}{\gamma_{mad}} Median\left(\left|w_{1,1}\right|\right) \tag{4}$$

$w_{1,1}$: is a WPC sequence of node (1,1)
The constant $\gamma_{mad} = 0{,}6745$ in equation (4) makes the estimate of median absolute deviation unbiased for the normal distribution.
nc : number of all the WPC of the ascending node index
cfs : content all the WPC of the ascending node index ($W_{5,0}, W_{5,1} \ldots W_{3,7}$)

$$thres = \left|cd(t)\right| \text{ where } t = 1 \ldots ncd \tag{5}$$

$thres$ contain absolute value of WPC stored in decreasing order, cd content the WPC of the ascending node index ($W_{5,1}, W_{5,2} \ldots W_{3,7}$) and ncd is the number of the WPC in the cd

$$A = cumsum(thres^2) \tag{6}$$

$cumsum$: compute the cumulative sum along different dimensions of an array

$$valthr = index\_\min\left(2\sigma^2 t\left(\alpha + \log\left(nc/t\right)\right) - A\right) \tag{7}$$

α : is a tuning parameter of penalty term ($\alpha = 6.25$)

$$Maxthr = \max\left(\left|cfs\right|\right) \tag{8}$$

$$Valthr = \min\left(valthr, Maxthr\right) \tag{9}$$

Where $Valthr$ denotes threshold value.

2.2.2 Thresholding algorithms

In this subsection, we review the most used thresholding algorithms, both hard and soft thresholding techniques proposed in [6] can be implemented to denoising speech signal. The hard thresholding function is defined for threshold λ as

$$\delta_\lambda^H(x) = \begin{cases} 0 & |x| \le \lambda \\ x & |x| \succ \lambda \end{cases} \tag{10}$$

in this thresholding algorithm, the wavelet coefficients x less than the threshold λ will be replaced with zero.
and the soft thresholding function is defined as

$$\delta_\lambda^S(x) = \begin{cases} 0 & |x| \le \lambda \\ sign(x)(|x| - \lambda) & |x| \succ \lambda \end{cases} \tag{11}$$

which can be viewed as setting the components of the noise subspace to zero, and performing a magnitude subtraction in the speech plus noise subspace. (figure 3)

2.2.3 Modified soft thresholding procedure

Each one of these algorithms defined above has its own disadvantages. The hard thresholding procedure creates discontinuities in the output signal is disadvantage, and in soft thresholding algorithm, the existence of the bias is the disadvantage. But soft thresholding procedure is near optimal for the signals corrupted by additive white Gaussian noise, however, some considerations applying the thresholding method (hard or soft thresholding method) to speech signal since the speech signal in the unvoiced region contains relatively lots of high frequency components that can be eliminated during the thresholding process. For improving these disadvantages, a modified soft thresholding (Mst) algorithm was been introduced and it is defined as follow [4] (see Figure 3):

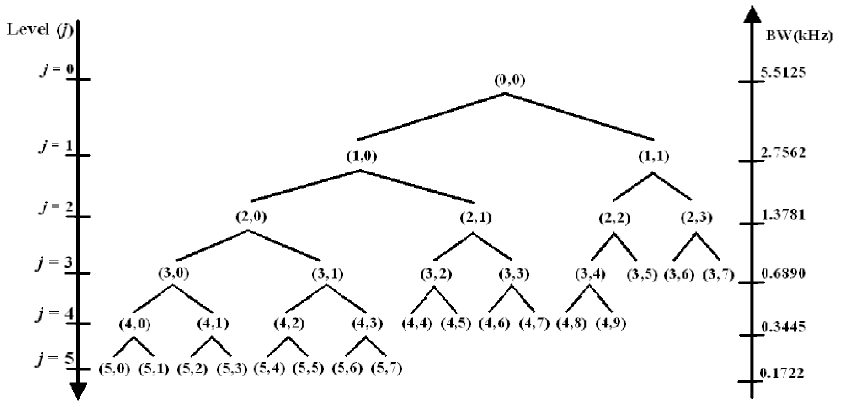


Figure 2: The tree structure of PWPT

$$y = \delta_{\lambda}^{Mst}(x) = \begin{cases} \theta x & |x| \prec \lambda \\ \operatorname{sgn}(x)(|x| + \lambda(\theta - 1)) & |x| \geq \lambda \end{cases} \qquad (12)$$

Where $x \in w_{j,i}$ and $y \in \overline{w}_{j,i}$ if $\overline{w}_{j,i}$ is the output column vector of denoised wavelet coefficient sequence. WPD subband i and level j as defined in equation (3). The inclination coefficient θ introduced in equation (12) is defined as follows:

$$\theta = \beta \frac{\lambda}{\max(w_{j,i})} \qquad (13)$$

β is the inclination adjustment constant. The main idea of modified soft thresholding is the introduction of the inclination coefficient θ, which prevents crudely setting to zero the wavelet coefficients whose absolute values lie below the threshold λ. The modified soft thresholding procedure is equivalent to the soft thresholding for $\beta = 0$. In our case the inclination adjustment constant β has been set to 0.5.

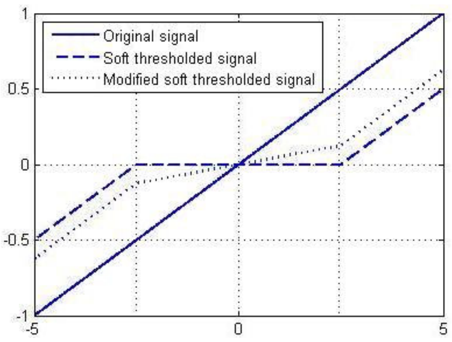


Figure 3: Characteristic of soft and modified soft thresholding technique, threshold λ is set to 0.5 in the figure above. In the case of modified soft threshold the parameter β is 0.5

2.3 Mel-frequency product-spectrum cepstral coefficients

On using the IPWPT, we obtained the enhanced speech signal $n\tilde{x}(n)$ and we compute the robust feature MFPSCC as described in [2]

The MFPSCCs are computed in the following four steps:

1) Compute the FFT spectrum of $\tilde{x}(n)$ and $n\tilde{x}(n)$. Denote them by $X(k)$ and $Y(k)$.

2) Compute the product spectrum

$$Q(k) = \max\left(X_R(k)Y_R(k) + X_I(k)Y_I(k), \rho\right) \qquad (14)$$

Where

$$\rho = 10^{\frac{\sigma}{10}} \cdot \max\left(X_R(k)Y_R(k) + X_I(k)Y_I(k)\right) \qquad (15)$$

σ is the threshold in dB (in our case $\sigma = -60dB$).

3) Apply a Mel-frequency filter-bank to $Q(k)$ to get the filter-bank energies (FBEs).

4) Compute DCT of log FBEs to get the MFPSCCs.

In all our experiments, the performances of ASR system are enhanced by adding time derivatives and log energy to the basic static parameters for different features. The delta coefficients are computed using the following regression formula

$$d_t = \frac{\sum_{\theta=1}^{\Theta} \theta (c_{\theta+1} - c_{\theta-1})}{2\sum_{\theta=1}^{\Theta} \theta^2} \qquad (16)$$

Where d_t is the delta coefficient computed in terms of the corresponding static coefficients $c_{t-\Theta}$ to $c_{t+\Theta}$. The same formula is applied to the delta to obtain acceleration coefficients.

3 Graphical comparison between the different features

Figure 4 shows a sample comparison between PNFR, Mfpscc and corresponding MFCC features for *Arabic digit one* obtained before DCT operation for different SNR levels. As standard in MFCC, a window size of 25 ms with an overlap of 10 ms was chosen, and cepstral features were obtained from DCT of log-energy over 22 Mel-scale filter banks. The degradation of spectral features for MFCC in the presence of white noise is evident, whereas PNFR obtained with soft thresholding (PNRF_soft) and Mfpscc features prevail at elevated noise levels. For SNR < 10dB we can see clearly that PNFR_soft is better noise robustness than mfpscc features.

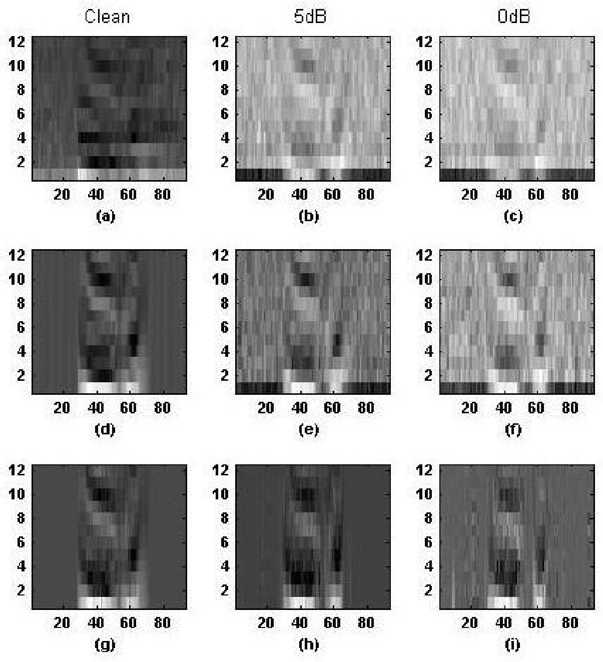


Figure 4: MFCC features (a)-(c), MFPSCC features (d)-(f) and PNRF_soft (g)-(i) for Arabic digit one, under different SNR conditions (clean, 10 dB and 0 dB).

4 Speech recognition experiments

In the experiments reported in this paper, isolated digit recognition experiments were performed using the Arabic digit corpus database from the national laboratory

of automatic and signals of University of Badji-Mokhtar Annaba Algeria, which were designed to evaluate the performance of automatic speech algorithms.
This database contains 90 speakers: 46 male and 44 female, each speaker repeats each Arabic digit 10 times. The leading and trailing silence is removed from each utterance. All samples are stored in Microsoft wave format files with 11025Hz sampling rate, 16 bit PCM, and mono-channels.

In our experiments, training is performed on clean speech utterances and testing data, which is different from the training data, is corrupted by different real-world noises added at the SNRs from -5 dB to 20dB at the step of 5dB, are used to evaluate the performance of a speech recognizer system. Four types of additive noises were used: white noise, pink noise, factory noise (plate-cutting and electrical welding equipment) and F16 cockpit noise selected from Noisex-92 database [15].

There are two test sets, In the test set A, There are 10 utterance of each digit (0-9) from each speaker (90 speakers): 6 of the utterance are for training and 4 remaining are for testing, what gives 5400 utterances for clean training and 3600 utterances were used for testing the system.

In the test set B, The training set contained 10 utterances of the Arabic digits each from 60 speakers (31 male and 29 female) comprising a total of 6000 utterances, and the test set contained isolated digits from 30 other speakers (15 male and 15 female) for a total of 3000 utterances.

A recognition system was developed using the Hidden Markov Toolkit (HTK) [10], implementing a 15 state left-to-right transition model for each digit where the probability distribution on each state was modeled as a three-mixture Gaussian.

We measured the robustness by comparing the word accuracies obtained with the proposed method and baseline feature parameters. As a baseline, the recognition system was developed using MFCC features comprising of 12 cepstral coefficients (0th coefficient is not used), log energy, delta and accelerator coefficients, totally 39 coefficients.

In the calculation of all the features, the speech signal was analyzed every 10ms with a frame width of 25ms multiplied with hamming window, accept proposed feature there is no need to apply Hamming window. The Mel filter bank was designed with 22 frequency bands in the range from 0 Hz to 5.51 kHz.
Tables 1 and table 2 show the accuracies obtained for various noise types with the different features. The last column is the average accuracy under different SNRs between clean and -5dB. From the results we may draw the following conclusions:

1. For clean speech, the performance of both features MFCCs and MFPSCCs are comparable, with high recognition rates. They provide better performance than the PNRF for the two test sets.
2. At SNR between 20 and 10dB, MFPSCC feature demonstrates much better noise robustness than other features for all noise types.
3. At SNR between 5 and -5dB the PNRF_soft features and PNRF with modified soft thresholding algorithm (PNRF_mst) obtain better performance than other features.
4. For white noise the PNRF_soft features obtain better performance than PNRF_mst, which indicate that the soft thresholding procedure reduces efficiently the level of additive white noise.
5. For pink, factory and f16 noises PNRF_mst features demonstrate significantly better performance than PNRF_soft features, which indicate that modified soft thresholding is better able to reduce the level of additive colored noise in the input speech signal.

Noise type	**Features set**	**SNR (dB)**							**Ave**
		Clean	**20**	**15**	**10**	**5**	**0**	**-5**	
White	**MFCC**	98.55	97.55	96.03	90.78	76.69	48.04	22.70	75.76
	MFPSCC	**98.61**	**98.33**	**98.08**	96.44	92.47	75.85	34.04	84.83
	PNRF_Mst	97.78	97.72	97.50	**96.75**	**92.89**	80.11	48.99	87.39
	PNRF_Soft	97.08	96.50	95.94	95.03	92.69	**85.72**	**65.05**	**89.71**
Pink	**MFCC**	98.55	96.55	91.94	80.30	61.79	35.76	16.00	68.69
	MFPSCC	**98.61**	**98.60**	**97.75**	**96.33**	91.05	71.13	42.01	85.06
	PNRF_Mst	97.78	97.42	97.22	96.17	**92.14**	79.41	49.37	**87.07**
	PNRF_Soft	97.08	96.69	96.05	94.72	91.33	**81.24**	**50.46**	86.79
Factory	**MFCC**	98.55	95.11	88.77	75.44	57.57	35.59	20.06	67.29
	MFPSCC	**98.61**	**98.59**	**97.36**	**95.94**	**90.28**	71.69	40.54	84.71
	PNRF_Mst	97.78	97.22	96.92	95.42	90.08	**77.10**	44.90	**85.63**
	PNRF_Soft	97.08	96.64	95.75	93.61	89.08	74.91	**45.23**	84.61
F16	**MFCC**	98.55	94.28	85.94	72.55	54.29	34.04	17.09	65.24
	MFPSCC	**98.61**	**98.60**	**97.17**	**94.69**	85.79	63.02	30.90	81.25
	PNRF_Mst	97.78	97.19	96.80	94.64	**87.97**	**68.38**	**37.09**	**82.83**
	PNRF_Soft	97.08	96.47	95.61	93.22	87.44	67.88	32.81	81.50

Table 1: Digit recognition accuracy (%) for different features of test set A (new speech samples from speakers whose speech was used for training system).

Noise type	**Features set**	**SNR (dB)**							**Ave**
		Clean	**20**	**15**	**10**	**5**	**0**	**-5**	
White	**MFCC**	**97.80**	96.77	95.03	88.93	74.49	43.91	18.34	73.61
	MFPSCC	97.60	**97.47**	**97.13**	**96.03**	92.13	77.33	39.41	85.30
	PNRF_Mst	97.00	96.87	96.67	95.47	**92.36**	80.83	49.65	86.97
	PNRF_Soft	96.27	95.67	95.07	93.73	91.00	**84.09**	**63.05**	**88.41**
Pink	**MFCC**	**97.80**	95.63	89.36	79.03	60.59	39.28	22.44	69.16
	MFPSCC	97.60	**97.59**	**97.07**	**95.50**	90.30	68.99	39.48	83.79
	PNRF_Mst	97.00	96.70	96.03	94.83	**90.43**	77.49	46.28	**85.53**
	PNRF_Soft	96.27	95.77	95.23	93.73	89.63	**78.09**	**48.92**	85.37
Factory	**MFCC**	**97.80**	93.93	87.16	73.12	54.52	35.88	22.71	66.44
	MFPSCC	97.60	**97.59**	**96.70**	**95.07**	**88.13**	69.66	37.35	83.15
	PNRF_Mst	97.00	96.37	95.47	93.70	87.86	**73.86**	42.08	**83.76**
	PNRF_Soft	96.27	95.33	94.43	92.23	86.76	73.02	**42.61**	82.95
F16	**MFCC**	**97.80**	92.63	83.59	69.26	50.72	34.41	19.87	64.04
	MFPSCC	97.60	**97.59**	**95.93**	**93.30**	82.22	58.82	29.68	79.30
	PNRF_Mst	97.00	96.40	95.63	92.86	**86.26**	**66.32**	**34.38**	**81.26**
	PNRF_Soft	96.27	95.53	94.70	91.86	85.06	66.12	31.78	80.18

Table 2: Digit recognition accuracy (%) for different features of test set B (speech samples from speakers whose speech was not used for training system).

5 Conclusion

In this paper we presented a novel speech feature extraction procedure, for deployment with recognition systems operating under various noise types and different levels of SNR. Results showed that The PNRF (PNRF_soft and PNRF_mst) features improved efficiently average recognition accuracy, especially at low SNRs level (-5 to 5dB). PNRF features give better performance than MFCC and MFPSCC features.

Acknowledgement

This work was supported by LASA laboratory of university Annaba algeria. The authors would like to thank Professor Mouldi Bedda for Helpful discussion.

References

[1] K.K. Paliwal and L. Alsteris, Usefulness of Phase Spectrum in Human Speech Perception, *Proc. Eurospeech*, pp. 2117-2120, 2003.

[2] D. Zhu and K. Paliwal, Product of power spectrum and group delay function for speech recognition, *Proc ICASSP 2004*, pp. I-125 I-128, 2004.

[3] Y. Gong, Speech recognition in noisy environments: A survey, *Speech Communication*, vol. 16, No. 3, pp. 261-291, 1995.

[4] B. Kotnik, Z. kacic and B. Horvat, The usage of wavelet packet transformation in automatic noisy speech recognition systems, *IEEE, Eurocon 2003*, Slovinia, vol. 2, No. 2, pp. 131-134, 2003.

[5] L. Birgé, P. Massart, From model selection to adaptive estimation, in D. Pollard (ed), Festchrift for L. Le Cam, Springer, vol. 7, No. 2, pp. 55-88, 1997.

[6] D. L. Donoho, De-noising by Soft-thresholding, IEEE Trans. Inform Theory, Vol. 41, No. 3, pp. 613-627, May 1995.

[7] D. L. Donoho, Nonlinear Wavelet Methods for Recovering Signals, Images, and Densities from Indirect and Noisy Data, Proceedings of Symposia in Applies Mathematics. Vol. 47, pp. 173-205, 1993.

[8] M. N. Stuttle, M.J.F. Gales , A Mixture of Gaussians Front End for Speech Recognition, *Eurospeech 2001*, pp. 675-678, Scandinavia, 2001.

[9] J. Potamifis, N. Fakotakis, G. Kokkinakis, Improving the robustness of noisy MFCC features using minimal recurrent neural networks, Neural Networks, IJCNN 2000, Proceedings of the IEEE-INNS-ENNS International Joint Conference on, vo1.5, pp. 271-276, 2000.

[10] S. Young, The HTK Book, Cambridge University Engineering Department, Cambridge, UK, 2001.

[11] B. Carneno and A. Drygajlo, Perceptual speech coding and enhancement using frame-synchronized fast wavelet-packet transform algorithms. IEEE Trans. Signal Process. 47 (6), pp.1622-1635, 1999.

[12] I. Pinter, Perceptual wavelet-representation of speech signals and its application to speech enhancement. Comput. Speech Lang. vol. 10, pp. 1-22, 1996.

[13] E. Zwicker, E. Tergardt, Analytical expressions for critical-band rate and critibandwith as a function of frequency. JASA68, pp. 1523-1525, 1980.

[14] M. Jansen, Noise reduction by wavelet thresholding. New York: Springer-Verlag, New York. 2001.

[15] A. Varga, , H. Steeneken, , M. Tomlinson, D. Jones, The NOISEX-92 study on the effect of additive noise on automatic speech recognition, Technical report, DRA Speech Research Unit, Malvern, England, 1992. Available from: http://spib.rice.edu/spib/select_noise

The Modelling of Manpower by Markov Chains – A Case Study of the Slovenian Armed Forces

Damjan Škulj, Vasja Vehovar and Darko Štamfelj
University of Ljubljana, Faculty of Social Sciences
E-mail: damjan.skulj@fdv.uni-lj.si, vasja.vehovar@fdv.uni-lj.si darko.stamfelj@fdv.uni-lj.si

Keywords: manpower planning, Markov chains, controlled Markov chains

Received: August 9, 2007

The paper presents a case study of manpower planning for the Slovenian armed forces. First, we identified 120 types of military segments (including civil servants). Next, administrative data were used to estimate the transitions between these segments for the 2001-2005 period. Markov chain models were then applied and 5-, 10- and 20-year projections were calculated. Various discrepancies were discovered between the projected structures of the military segments and the official targets. Finally, we addressed two optimisation questions: 'Which transitions would lead to the desired structure in five years' time?' and 'Which transitions would sustain the desired structure in the long run?'. A heuristic modelling approach was applied here, i.e. we used a simulation model with a specific loss function. To perform feasible simulations of the probabilities in a 120 x 120 matrix, only those transitions were simulated where experts had previously estimated that real measures existed to provide potential change (recruitment policy, regulation of promotion, retirement strategy etc).

Povzetek: V članku je predstavljen primer načrtovanja kadrov v Slovenski vojski.

1 Introduction

Efficient manpower planning is a crucial task of managing large organisations such as transportation or industrial corporations, the state administration or military systems. All of these systems comprise many segments of employees with specific roles and job descriptions. Skills needed to perform assigned tasks are usually acquired through special training or long work experience. Both a shortfall and a surplus of skilled staff can be costly and very inefficient. To prevent such difficulties, the future needs of personnel have to be predicted well in advance, while corresponding strategies to achieve the desired structure must be adopted.

Knowledge about these processes is important for predicting the future development of the manpower structure in complex organisations. In large systems, such predictions are usually based on previous experience. However, knowledge gained from such experience is often difficult to apply without appropriate mathematical or statistical models and corresponding computational tools.

Pending on the goals, various mathematical models can be applied in manpower planning. Obviously, the problems cannot be fully addressed by only using tools of the spread-sheet type. The choice depends on many factors, such as the size of the system being analysed, available knowledge of the processes that govern the system structure's dynamics, the methods available to control the processes and the ability to predict the consequences of actions concerning regulations. Moreover, the choice of the appropriate model often depends on its complexity. While complex models can supply very accurate results, they often require data that are not easy to collect, or parameters that may only be vaguely known, especially if a very large number of them have to be specified. Consequently, the reliability of the resulting outputs is then put in question. For very large systems, simpler and more robust models are therefore often a better choice. A good overview of the existing models used in workforce planning can be found in Wang (2005). For other references, also see Grinold and Marshall (1977), Price et al. (1980), Purkiss (1981), and Georgiu and Tsantas (2002).

The most basic information that can be used to model manpower dynamics is the rate of transitions between different segments of the system, i.e. the transition probabilities. Transitions are usually consequences of either promotions, transfers between assignments or wastage and input into the system. Often transitions are controlled by certain rules that govern the system and cannot be arbitrarily changed. If this is the case, planning has to be especially careful since slight changes in policies can have considerable consequences on the future development of the manpower structure.

In several cases, the models used to predict the future structure of a dynamic system are based on Markov chains and their derivatives, such as semi-Markov chains. Both are based on the assumption that the rules governing the system's manpower dynamics do not change very often and that future dynamics will follow patterns observed in the past. While classical Markov chains view segments as homogeneous, semi-Markov chains additionally involve the time a person has spent in a segment, of course at the cost of the model's simplicity and therefore the possibility to reliably estimate its parameters. A thorough description of many variations of Markov and some other manpower planning models can be found in Bartholomew et al. (1991), Vassiliou (1998) or Vajda (1978). Besides Markov models, other approaches to the problem are also possible, such as models based on simulations or system dynamics models (Wang, 2005). Applications of manpower models used in the specific case of military manpower planning can be found in Jaquette et al. (1977), Murty et al. (1995), Smith and Bartholomew (1988).

In our particular case of modelling the structure of the Slovenian armed forces, the number of segments alone was relatively large. Together with other potential problems related to the data collection, these were the main reason against using the more complex semi-Markov chains. Moreover, transitions between segments are surprisingly complex where, besides recruitment, promotions and wastage from the system, many more transitions to other segments also occur such as transitions from military to administrative positions and vice-versa.

Models based on Markov processes can be divided into the following groups, depending on the level of the structural control i.e. the ability to attain and maintain the desired structure:

1 *Descriptive-predictive models.* This group is mainly concerned with the development and analysis of a time homogeneous Markov model whose parameters are often based on historical observations. It can be used to predict the behaviour of a system in time. The models in this group have no intention to search for any kind of optimal control, but only give descriptions and forecasts. Several models of this group can be found in Bartholomew et al. (1991) or Price et al. (1980). In this paper, we use such a model in our first part to make predictions on the future development of the system.

2 *Control theory models-normative models (Markov decision processes).* This group tries to find optimal set of policies in order to minimise certain loss functions such as the cost of recruiting new workers or maintaining the existing structure. The basis of these models is the work of Howard (1960) and they can be found in Grinold and Stanford (1974), Zanakis and Maret (1981), Lee et al. (2001) or a very general treatment can be found in Li et al. (2007). Our approach to searching for the transitions leading to a desirable structure could be regarded as belonging to this class, although its key concern is to find any satisfactory policy rather than to select a particular policy satisfying some additional optimality conditions.

While in the first part we use the classical descriptive-predictive model, the structural control part of the problem partly belongs to the class of control theory problems, although it has a very specific form and thus does not fit well in the context of controlled Markov chains or related models. The main specific point is that the choice of feasible policies is severely restricted by the Ministry of Defence and the problem, at least at this stage, is thus not in finding a policy that would satisfy some additional optimality criterion, but rather in finding *any* policy that leads to the desired structure of the system. A similar problem was theoretically studied by Antončič (1990), but subject to less severe restrictions. We thus claim that our particular problem in fact belongs somewhere between the first and second groups of the above models, and that it requires some kind of a specific treatment.

The problem we are dealing with thus mainly consists of two parts. In the first part we identify relevant segments and the transitions between them. The goal is to make predictions of the future sizes of the segments if the current transitions ruled the system's dynamics. This would likely be the case if no further regulations were implemented. In the second part we study the attainability and maintainability problem. We first identify transitions that could be regulated by the Ministry of Defence by setting appropriate measures and policies. Then we tried to modify them within the limits given by the Ministry in order to achieve the required structure. The problem is complex mainly because transitions that are not controllable represent a substantial part of the system's dynamics and are therefore the main cause of the large discrepancies between the projected and the required structures. Being ruled by mechanisms unknown to us, in the best case we can assume that they will remain roughly the same in the following years. To achieve the required structure, we must then appropriately set the controllable transitions.

The estimated probabilities of the uncontrollable transitions are used as expectations of future transition probabilities and thus we effectively assume a deterministic model with two classes of transitions – fixed and controllable. The only objective at this stage is to find a sufficient set of transitions that would lead to the required structure. Among the models found in the existing literature none of them seemed quite appropriate for resolving this particular problem, although a modification of some existing analytical model to fit our framework is a promising direction for our further work. However, at this stage the approach using computer-based simulations proved to be sufficiently effective in producing satisfactory results. The idea is therefore to simulate a large number of randomly generated scenarios and pick those yielding a satisfactory structure after a given period. However, the implementation depends to a large degree on the particular specifics.

The practical implementation of the approach to solving our problem also contains a web-based user application that was developed to allow non-mathematicians to change the parameters and analyse the consequences. The application's user interface is designed to be both user-friendly and flexible so that it allows practically unlimited possibilities in testing different scenarios. The results of testing are displayed simultaneously in real time. The application is available on-line and no programmes other than web browsers are needed to run it.

The prediction part of the model is thus made in a fully interactive manner; however, the structural control part is still too complex to be implemented by ordinary users, especially because of the computational complexity which requires several manual adjustments during optimisation. In addition, the process of preparing the data to estimate the parameters is technically very demanding because it requires the combining of several software tools and is therefore not available in an automated form.

The paper has the following structure: Section 2 contains a description of the method used with some mathematical background, Section 3 describes implementation of the method for calculating projections of the manpower structure of the Slovenian armed forces and the administration of the Ministry of Defence, while Section 4 presents the results. Conclusions are presented at the end.

2 Description of the method

2.1 Basic model

The model used for manpower planning in the Slovenian armed forces and the administration of the Ministry of Defence is based on Markov chains. The description of the mathematical background of this and related models can be found in Bartholomew et al. (1991), Vassiliou (1998), and Grinold and Marshall (1977). The usual assumption of those models is that the system modelled consists of clearly defined segments and that the transitions between them are time-homogeneous and independent of history. To satisfy these requirements, the system must be sufficiently large to diminish the effect of random variations in time allowing transitions to be analysed on the aggregate level.

Markov chains are used to model random processes evolving over time. A crucial assumption used is the Markov property which is best described by saying that the process is "memoryless", which means that its future states only depend on the present state rather than its history. Another assumption which is usually posed is that transition probabilities are time-homogeneous, which means that they are independent of time. Of course, these requirements are often not entirely fulfilled; however, omitting them would result in more general non-homogeneous models (see Vassliou (1981, 1998), Gerontidis(1994)) that require additional data to estimate the parameters.

A sequence of random variables $X_1, X_2, ..., X_n$, ... is a Markov chain if every variable can assume any value from a set $S = \{s_1, s_2, ..., s_m\}$, whose elements are called states. A particular realisation of the process is then a sequence of values from S. Mathematically we can describe the Markov property by requiring that $P(X_{n+1} = s_j \mid X_n = s_{i,...}, X_1 = s_k) = P(X_{n+1} = s_j \mid X_n = s_i) = p^{(n)}_{ij}$, where n denotes the time points. This means that the probability that the process will be in state s_j at time n+1, given that it is in state s_i at time n, is $p^{(n)}_{ij}$. If in addition $p^{(n)}_{ij=} p_{ij}$, for every n, the chain is homogeneous because the transition probabilities do not depend on time. The matrix $P=(p_{ij})$, whose entries are transition probabilities, is called a transition matrix.

In the particular case of manpower modelling, states are the segments of the system, and transition probabilities are understood as relative frequencies. Thus transition probability p_{ij} is interpreted as the ratio of persons in segment s_i at the time n that will make a transition into state s_j by the time n+1. Let γ^n be a statistical distribution over the set S, thus, γ^n_i is the number of persons in state s_i. On the assumption of transition matrix P, the distribution at the next time point is obtained by multiplying vector γ^n by P, $\gamma^{n+1} = \gamma^n P$. Thus, given the initial distribution and transition matrix we can predict the distribution at the next time point. Consequently, by repeating the same argument predictions for all future time points are enabled. The time interval used in our model is one year and so if a person makes more than one transition within a year this is recorded as a single transition.

The actual model used is based on the model described in Bartholomew (1991), where in addition a vector of recruits $\boldsymbol{r}$ is added. The model also allows wastage $\boldsymbol{w}$, which accounts for those people who leave the system. According to this model, the transition matrix P need not be stochastic but only substochastic, i.e. the sum of rows may be less than 1, where the difference is wastage $\boldsymbol{w}$, thus,

$$w_i = 1 - \sum_{i=1}^{m} p_{ij}.$$

We assume a model with a constant recruitment vector since the only recruitment to the system is births in the general population. The actual recruitment to military segments is then modelled as transitions between general and military segments. Thus the model used takes the following general form:

$$\boldsymbol{\gamma}^n = P\boldsymbol{\gamma}^{n-1} + \boldsymbol{r}.$$

Hence, the military segments used in the model are regarded as a subset of segments of the general population. The reason for this is that new employees are recruited from general segments whose size then determines the number of possible new recruits every year. So, for instance, diminishing generations of school children due to the lower birth rates seen in recent years affect the number of potential candidates to enter military service. For this, we assume that the proportion of them interested in such a career is roughly constant.

The whole population of the inhabitants of Slovenia was divided into segments relevant to the model. General population is, for instance, divided into the following six segments: non-active population, which mainly consists of pre-school and primary school children; secondary school pupils, students, working people, unemployed, and retired persons. Military segments are treated separately from the general population because their relatively small size does not affect the dynamics of the general population.

The total number of military segments is 120, which makes the total number of segments equal to 126. Of course, this is not the only possibility to model our segments. If the general segments were omitted, for instance, the same effect would be achieved by adequately modifying the recruitment vector. The changes in the size of the military segments are consequences of the following factors:

1 transitions between segments;
2 wastage from the system (retirement…);
3 input from the general population (recruitment); and
4 input to the general population, which is modelled as births.

Because of the relatively small size of the military segments, wastage from the army may be neglected and treated as wastage from the system even though actually there may also be transitions to other segments of the general population. The model assumes a constant level of annual input to the system, which we assume is the current birth rate in Slovenia. Slight deviations from this assumption in the following years would not affect the model substantially. The size of the whole population follows general trends in society.

Recruitment into military segments is modelled by transitions from the general to the military segments. The model assumes that those transitions are regulated by a set of rules that do not change often. We adopt the assumption that the transitions will remain similar to those observed in the past if regulations do not change. If, for instance, 10% of officers in a segment were promoted to a higher rank, we assume that this will continue in the following years. This assumption is, of course, sometimes questionable because it is rarely true that all members of a segment have the same probability of being promoted. However, the available data did not allow the modelling of any heterogeneity within the segments.

The model involves a mixture between the deterministic and stochastic approaches. We initially start with stochastic transitions estimated on the base of historical transitions. These are then mixed with those transitions that are possible to regulate and are thus deterministic. Another model with a mixture of stochastic and deterministic transitions can be found in Guerry (1993), where only wastage transition probabilities are modelled as stochastic while promotion and demotion probabilities are considered constant.

2.2 Attainability and maintainability

The next step after making predictions based on the model is to exercise structural control i.e. to find transitions leading to the desired structure after a certain time period, or at least close to it. Thus we are solving the attainability problem. Methods of solving attainability problems have been discussed in the literature (see e.g. Davies (1973) or Nilakantan and Raghavendra (2005)), but none of the approaches seems to completely resolve our specific problem, where the task is to be accomplished by resetting a certain subset of transitions within given constraints. This is because not all transitions can be controlled, while those that can be controlled are not allowed to be set entirely arbitrarily. Therefore, it is crucial to determine which transitions can be changed and to what degree. Once this is determined, the transitions yielding the best possible result are to be found. We find a sufficient solution using computer simulations. The idea behind the simulations is to generate many possible scenarios by varying the allowed transitions within the allowed intervals and selecting the most suitable ones.

A sufficient solution, that is the set of transitions leading close enough to the aimed structure, is thus sought within the limits that can be implemented. We thus first had to find out which transitions are possible to regulate and to what extent. We acquired this information in co-operation with the Ministry of Defence, which provided us with a set of all transitions that can be controlled and typically the upper bound for each transition in terms of the maximum ratio of employees from each segment that can make a transition to another segment. Typically, the transitions that can be controlled are promotions. The transitions capable of sufficiently improving the resulting structure were then sought within these bounds.

About 500 such transitions were identified and lower and upper bounds for the transition probabilities were given. To find optimal transitions within these bounds, several approaches seem possible. One option is to decrease transitions to segments with an excess of units and increase those to segments with deficiencies. However, changing a single transition also affects other segments and the effects are difficult to predict beforehand. Thus it is difficult to determine which transition probability has to be changed to obtain the desired result. Also, such an iterative algorithm would to a large extent depend on various assumptions and features of a particular example. It would be reasonable to expect good results with such an approach in a simpler model where the usual transitions are promotions and recruitment only; however, this was not the case with our model. Because of the large number of other possible transitions this was unsuitable for our model. The other natural approach is to try to find an acceptable solution by systematically examining all possible combinations of transitions within a given range. However, a simple calculation showed that such an approach would be impossible to implement in such a large model.

A remaining possibility is Monte Carlo simulations where, instead of changing transitions systematically, this is done by randomly changing them within a particular amount. More precisely, the change of a particular transition was obtained as

$$p_{ij}^{new} = p_{ij}^{old}\left(1 + d_1 r_1\right) + d_2 r_2,$$

where the symbols denote

d_1 and d_2 are the factors denoting the maximum amount of the change and

r_1 and r_2 are random numbers uniformly distributed over the interval [0.5, 0.5].

Further, we required that all the changes sum to 0 so that wastage remains as estimated, since it is an uncontrollable parameter. Thus, in every step some transitions are picked and varied randomly, where d_1 and d_2 are parameters of the algorithm, as well as the number of chosen transitions. The process is iterative. If the matrix obtained in this way leads to a better result, this new matrix is used again in the next iteration. Some tuning is, of course, necessary such as determining the number of transitions to be changed at each time and the optimal values of the parameters, which depend on the particular instance. So far we have not done any systematic analysis of the effects of the parameters on the efficiency of the algorithm.

It soon turned out that it is impossible to obtain an exact solution leading to the satisfactory structure. This is due to many factors such as the set of transitions we are allowed to change and the amount of changes we are allowed to make for a particular transition and especially the short time available to reach the desired structure. Therefore, we have to satisfy ourselves with a solution sufficiently near to the required one. The concept of the best possible solution is thus not unique but depends on more or less subjective criteria. To illustrate this, suppose we have two segments with 50 and 500 units targeted, respectively. Is it better to end up with 60 units in the first and 490 in the second one, or is it better to have 50 units in the first one and 550 in the second? Clearly, this depends on what we are trying to achieve. If the main goal is that absolute deviations from the required sizes of the segments are as small as possible, the first scenario is better; however, those in the segment of size 50 are likely to have more specialised working tasks and therefore it is probably more important to have a better result in this segment. In the sense of relative deviations from the target, the second possibility is clearly better. Thus, another possibility would be to minimise relative deviations from the optimal structure. But this would clearly be again suboptimal. It is difficult to say in general where in the middle is "the right" criterion.

We have some general requirements of optimality, such as:

- Big deviations in one segment are less satisfactory than a large number of smaller ones, even though they sum equally.
- In larger segments bigger absolute deviations are allowed than in smaller ones.

A criterion satisfying the above requirements is expressed through a mathematical function. A function satisfying those criteria is, for instance, the following one:

$$\sum \frac{(x_i - x_{i0})^2}{x_{i0} + C},$$

In the above function the symbols denote the following:

- x_i: the size of the ith segment as a result of a given scenario;
- x_{i0}: target size of the ith segment; and
- C: a constant – varying this constant changes the relative importance of either segments with a smaller number of units or those with a larger number of units. A value that turns out as balanced in our particular case is C=50.

The above loss function may be understood as a distance function measuring the distance between the solution achieved in a given scenario and the optimal solution. The algorithm for attaining the desired manpower structure thus tries to minimise this distance.

Once the required structure is attained, the task is to find a strategy to maintain the obtained structure. Our goal is to solve it in a similar manner as the problem of attainability; that is, by setting the set of controllable transition to values that would, at least approximately, maintain the obtained required structure. Here too, simulations do the task sufficiently. The only modification from solving the previous task is that we set the initial and the target structure to be the same and then, using simulations, try to find transitions that preserve the structure. Effectively, we are thus seeking a transition matrix within an allowed set of matrices whose stationary distribution is the required distribution. We therefore call the resulting transitions stationary transitions.

3 Preparation of the data

3.1 Input data and population segments

The first phase of the analysis included the identification of segments from the available data. The military segments were identified on the basis of the administrative titles of employees, by regarding employees holding the same administrative title as members of the same military segment. The Ministry provided us with anonymised data (such as identification number, period of employment, administrative title, education etc.) for all employees in the Slovenian armed forces and the administration of the Ministry of Defence in the period between 1 January 1997 and 31 August 2006.

The most demanding task was estimating the transitions between the segments. They were calculated on the basis of past transitions identified from the data on employees in the Slovenian armed forces. This was possible thanks to the accurate manpower data of the Ministry of Defence.

In the data, every employee's transition from one segment to another and every extension of the employment contract in the same segment was registered as a new entry with the same identification number, and a new initial and final date of their employment contract. From that, transitions between various segments were calculated for every year in the period. Also, the number of people in each segment as at 30 June 2007 was obtained as a basis for calculating projections (as the initial manpower structure).

3.2 Standardisation of the data

The model requires identifying the employees' administrative titles (i.e. affiliation to military segments) for each year in the given period for which data are available. An employee's administrative title usually does not change more than once a year. Therefore, the administrative titles were identified on 31 December for every year in the period between 1 January 1997 and 31 August 2006. The data enable the identification of administrative titles on this date for the period between 1997 and 2005.

However, the data did not enable us to identify from which of the six segments of the general population employees of the Ministry of Defence came from. It was also impossible to identify into which of the segments of the general population they went after they left the Ministry. For this reason, another segment called "outside" was introduced, which stands for the entire general population. Thus, transitions from the "outside" to military segments therefore denote new employments, while transitions to the segment "outside" from other segments denote deaths, retirements or terminations of an employment contract. Consequently, transitions between military segments and specific segments of the general population were estimated so that their cumulative effect at the aggregate level was equal to transitions between military segments and the segment "outside".

In the next phase, transitions between segments of every individual employee in the Slovenian armed forces and the administration of the Ministry of Defence were identified on the basis of comparisons between his or her segment in two consecutive years on 31 December.

Yet it is possible that a person made a transition more than once a year. In that case, all those transitions were only registered as a transition between the starting and ending segment since the person's segments were only checked on 31 December.

3.3 Estimating the transition matrix

For each year, the number of people in every segment and the number of transitions from one segment to another were counted. The number of transitions from a particular segment to another one was divided by the number of people in the starting segment. This gave us the transition probabilities between the segments for each year in the period. The average transition probability between two segments in the period was calculated as a weighted mean:

$$\bar{x} = \frac{\sum_{i=1}^{n} w_i x_i}{\sum_{i=1}^{n} w_i},$$

where x_i denotes the transition probability in a certain year, and w_i the number of people at the start of the same year.

However, the transitions have changed substantially in recent years due to changes in legislation and the process of professionalising the Slovenian armed forces. Consequently, some of the transitions that were possible in the past are no longer possible today, and the transition probabilities have also changed. For those reasons only the data for the four most recent years, that is for the period from 31 December 2001 to 31 December 2005, were used to calculate the average transitions. Also in order to avoid such problems, the Ministry of Defence reviewed the transition matrix and marked those transitions that are no longer possible. The latter were then assigned the value 0 in the transition matrix, while the transitions on the diagonal (standing for the share of people who stay in the same segment each year) were correspondingly corrected so as to keep the same total proportion of people remaining in the system. The final result was transition matrix P with entries p_{ij} which summarise the proportion of employees of segment i that yearly make the transition to segment j.

The next step is to identify those transitions that can be regulated by the Ministry of Defence's personnel departments. By simulating changes in those transitions it is possible to identify a transition matrix that leads to the desired manpower structure in a certain period. The transitions that can be regulated are, for example, promotions and new employments. What we want to know, for instance, is the optimal share of people that can be promoted from one segment to another, and the optimal number of people that can be employed within the segment each year.

4 Results

4.1 Basic model

The results of our simulations show how the manpower structure will change over the course of time after 2006 if the future transitions are equal to the average transitions in the period from 31 December 2001 to 31 December 2005. A comparison of the projected manpower structure in 2010 and the desired structure also shows which military segments will face manpower shortages or surpluses.

The projected number of people in seven selected military segments (out of 120) in the period from 2007-20011 and in 2027 according to a continuation of the transitions observed in the period from 31 December 2001 to 31 December 2005, the desired manpower structure in 2010 and differences between the desired and projected structure in 2010[1] are presented in Table 1.

As can be seen from Table 1, in this scenario the manpower projection for 2010 differs considerably from the desired manpower structure: some military segments (like S1, S3, S4 and S5) will face a manpower shortage, while others (like S2, S6 and S7) will face manpower surpluses. The deviations are large in both absolute and relative numbers (i.e. an absolute deviation in relation to the desired number of employees in the selected military segment).

[1] The table only contains numbers for the selected seven military segments since the data for all of the segments cannot be published due to data protection reasons. The names of the segments were also encoded for that reason.

Table 1: Manpower structure for seven selected segments by the continuation of average transitions 2001-05

Year	Sg1	Sg2	Sg3	Sg4	Sg5	Sg6	Sg7
2007	51	944	700	125	33	488	635
2008	48	966	594	108	39	520	657
2009	46	973	505	97	46	551	692
2010	45	968	430	89	54	584	735
2011	43	953	368	84	61	619	781
2027	25	495	135	133	257	1267	1214
Desired 2010	67	853	760	145	81	573	642
Difference (proj. 2010 - desired 2010)	-22	115	-330	-56	-27	11	93
Difference in %	-33%	13%	-43%	-39%	-33%	2%	14

4.2 Attainability

Structural control was exercised on the basis of a transition matrix containing average transitions between 31 December 2001 and 31 December 2005, and the manpower structure of the Slovenian armed forces and the administration of the Ministry of Defence on 30 June 2007. The target year for achieving the desired structure of 2010 specified by the Ministry of Defence was reset to 2011. By doing so, an additional year was added. Otherwise, the period for attaining the desired structure would have been too short. The improvements due to the structural control are measured in terms of the change in the loss function described in Section 2.2. It turned out that about 1,000 transitions could occur and 500 of those are controllable.

If the transitions in the period from 2007 to 2011 were on average the same as those in the period from 31 December 2001 to 31 December of 2005, the value of the loss function would be equal to 3,133[2]. The method of attaining the desired manpower structure was implemented in the following way. For each iteration five transitions were randomly chosen among the controllable transition to be modified using equation (1), where optimal values of r_1 and r_2 turn out to be 0.05 and 0.01, respectively. If the resulting matrix leads to a more desirable structure, the following iteration uses the new matrix as the initial one. After 200,000 iterations the value of the loss function stabilises at 283, which clearly indicates a significant improvement. Increasing the number of iterations does not yield a much greater improvement.

Table 2: Attainability of manpower structure for seven selected segments (optimised transitions)

Year	Sg1	Sg2	Sg3	Sg4	Sg5	Sg6	Sg7
2007	51	944	700	125	33	488	635
2008	56	909	707	123	44	511	595
2009	59	876	713	127	54	528	588
2010	60	845	720	133	66	543	601
2011	61	816	728	140	77	558	624
2027	42	519	1015	228	294	890	900
Desired 2010	67	853	760	145	81	573	642
Difference (proj. 2011 - desired 2010)	-6	-37	-32	-5	-4	-15	-18
Difference in %	-9%	-4%	-4%	-3%	-5%	-3%	-3%

The projected number of people in selected military segments in the period from 2007 to 2011 and in 2027 by the continuation of transitions for attaining the desired structure, the desired structure in 2010, and differences between the desired structure in 2010 and the projected structure in 2011 are presented in Table 2.

Small differences between the desired structure and that achieved with the optimised transitions are also clearly seen in Table 2. The deviations are small in both absolute and relative numbers: none of the selected military segments will have a manpower shortage of

[2] Smaller values are better.

more than 37 people (S2) or 9% (S1), respectively. In Table 1, on the other hand, the projected number of employees in the segment (S3) with the largest deviation from the desired manpower structure faces a manpower shortage of 330 people or 43% of the desired number of employees.

4.3 Maintainability

The solution to the attainability problem described above provides a set of transitions that lead to a structure sufficiently close to the desired structure in a few years. It must be stressed that the current structure is quite far from optimal, which mainly reflects the fact that it transformed substantially in recent years due to the change from a conscript to a professional service. Therefore, to achieve the desired structure in only a few years the needs for recruitment and promotions tend to be higher than needed to merely sustain the structure once the desired structure is reached.

The next important step in exercising structural control is to find the transitions needed to maintain the desired structure after it has been reached. The method of finding such transitions is substantially the same as in the previous case. We take as an initial distribution the desired distribution and identify transitions, using simulations, so that they would preserve the structure. Even though the problem seems easier than the previous one, it turns out that it is impossible to sustain the structure in all of its parts. This indicates that some transitions prevent the system's stability. Identifying those transitions and examining ways to improve the stability is one of the tasks that still has to be accomplished.

The projected number of people in the selected military segments in the period from 2011 to 2015 and in 2031 according to the continuation of stationary transitions, the desired manpower structure in 2010 and differences between the desired structure in 2010 and the projected structure in 2031 are presented in Table 3.

Unlike in Tables 1 and 2, the projected manpower structure presented in Table 3 was calculated using the projected manpower structure for 2011 in Table 2 as the initial structure instead of the manpower structure on 30 June 2007. As can be seen, the projected manpower structure of the selected segments only slightly deviates from the desired one, even in 2031: the greatest manpower shortage does not exceed 26 people or 5%, respectively (S6).

Table 3: Maintainability of manpower structure in seven selected segments in a long run

Year	Sg1	Sg2	Sg3	Sg4	Sg5	Sg6	Sg7
2011	61	816	728	140	77	558	624
2012	61	816	728	140	77	556	621
2013	61	816	728	140	77	555	617
2014	62	816	728	139	77	553	615
2015	62	816	728	139	77	551	612
2031	62	813	716	135	77	532	599
Desired 2010	67	853	760	145	81	573	642
Difference (proj. 2031 - proj. 2011)	1	-3	-12	-5	0	-26	-25
Difference in %	2%	0%	-2%	-4%	0%	-5%	-4%

5 Conclusion

In this paper we presented a case study of applying a Markov chain model in the Slovenian armed forces. First, extensive administrative effort was needed to obtain data on the individual status of all employees for the 2001-2005 period. We needed an exact assignment (of each person and for each year) to one of 120 key military segments (soldier, lieutenant etc.). This then formed the basis for calculating transition probabilities in a 120 x 120 matrix. The Markov chain model was then applied to these data. Considerable gaps were found in the projected sizes of the segments compared to the official targets. Of course, experts and decision-makers were roughly aware of these discrepancies but the results of the modelling provided much more explicit and elaborated evidence of the problems related to future trends.

However, the Markov chain model itself could not provide an answer as to how to achieve the desired manpower structure. This problem was successfully addressed here by simulations. The simulation algorithm selected a solution closest to the target structure from a large number of computer-generated scenarios. A specific loss function was developed for this problem.

The approach described in this paper can be upgraded in several ways. For planning a smaller number of selected segments a semi-Markov model could be developed in which the "age" of the units in the segments (i.e. the time a person is employed in the segment) is considered in calculations of more accurate transition probabilities. Another direction is the implementation of a time non-homogeneous model instead of a time homogeneous model; this means that the transition probabilities could vary over time.

The accompanying web application that automated the above described calculation of manpower projections also demonstrated to be a very useful tool. The application used the existing manpower structure and some type of (assumed) transition matrix to calculate statistics and trends for past years and projections for the future.

Acknowledgement

We would like to thank the Slovenian Army for supporting this project and providing all the required help. We would also like to thank the four anonymous referees for their many useful suggestions and comments.

References

[1] Antončič, V.: Enodobni planski račun za velikost in sestavo, *Metodološki zvezki*, Vol. 7, 1990, p. 118.

[2] Bartholomew, D.J., Forbes, A.F., McClean, S.I., *Statistical techniques for manpower planning*, John Wiley & Sons, 1991.

[3] Davies, G. S.: Structural Control in a Graded Manpower System, *Management Science*, 1973, Vol. 20(1), Theory Series, p. 76.

[4] Georgiou, A.C., Tsantas, N.: Modelling recruitment training in mathematical human resource planning, *Applied Stochastic Models in Business and Industry*, John Wiley & Sons, Ltd. 2002, Vol. 18 (1), p. 53.

[5] Grinold, R.C., Stanford, R.E.: Optimal Control of Graded Manpower System, *Management Science*, 1974, Vol. 20 (8), Application Series, p. 1201.

[6] Grinold, R.C., Marshall, K.T.: *Manpower Planning Models*, North-Holland Pub Co, 1977.

[7] Gerontidis, I.I.: Stochastic Equilibria in Nonhomogeneous Markov Population Replacement Processes, *Mathematics of Operations Research*, 1994, Vol. 19(1), p. 192.

[8] Guerry, M.A.: The probability of attaining a structure in a partially stochastic model, *Advances in Applied Probability,* 1993. Vol. 25(4), p. 818.

[9] Howard, R.A.: Dynamic Programming and Markov Processes, The Technology Press of MIT and John Wiley & Sons, Inc., New York, London, 1960.

[10] Jaquette, D.L., Nelson G.R., Smith R.J., An Analytic Review of Personnel Models in the Department of Defense. 1977, RAND.

[11] Lee, H.W.J., Cai, X.Q., Teo, K.L.: An Optimal Control Approach to Manpower Planning Problem, *Mathematical Problems in engineering*, 2001, 7, p. 155.

[12] Li, Y., Chen, J., Cai X.: An integrated staff-sizing approach considering feasibility of scheduling decision, *Annals of Operations Research*, 2007, Vol. 155 (1), p. 361.

[13] Murty, K.G., Djang P., Butler W., Laferriere R., The Army Training Mix Model. *Journal of the Operational Research Society,* 1995. Vol. 46(3): p. 294.

[14] Nilakantan, K., Raghavendra, B. G.: Control aspects in proportionality Markov manpower systems, *Applied Mathematical Modelling,* 2005, Vol. 29(1): p. 85.

[15] Nilakantan, K., Raghavendra, B. G.: Length of service and age characteristics in proportionality Markov manpower systems, *IMA Journal Management Mathematics,* 2008, Vol. 19, p. 245.

[16] Price, W.L., Martel A., Lewis K.A.: A Review of Mathematical Models in Human Resource Planning. OMEGA, 1980. 8(6): p. 639.

[17] Purkiss, C.: Corporate Manpower Planning: a review of models. *European Journal of Operational Research,* 1981. 8(4): p. 315.

[18] Smith, A.R., Bartholomew, D.J.: Manpower Planning in the United Kingdom: An Historical Review, *Journal of the Operational Research Society*, 1988, Vol. 39(3), p. 235.

[19] Vajda S.: *Mathematics of Manpower Planning,* John Wiley & Sons, Chichester, 1978.

[20] Vassiliou, P.-C. G.: On the Limiting Behaviour of a Non-homogeneous Markov Chain Model in Manpower Systems. *Biometrika*, 1981, Vol. 68, p. 557.

[21] Vassiliou, P.-C.G.: The Evolution of the Theory of Non-Homogeneous Markov Systems. *Applied Stochastic Models and Data Analysis*, 1998. 13: p. 159.

[22] Wang, J.: A Review of Operations Research Applications in Workforce Planning and Potential Modelling of Military Training, DSTO Systems Sciences Laboratory, Edinburgh Australia, 2005.

[23] Zanakis, S.H., Maret M.W., A Markovian Goal Programming Approach to Aggregate Manpower Planning. *Journal of the Operational Research Society*, 1981. 32(1): p. 55.

Content-Based Watermarking for Image Authentication Using Independent Component Analysis

Dr. Latha Parameswaran
Professor, Department of Computer Science & Engineering,
AMRITA University, Coimbatore – 641 105, India
E-mail: lathapcp@yahoo.co.in

Dr. K. Anbumani
Former Director, Karunya School of Computer Science and Technology,
Karunya University, Coimbatore –641114, India
E-mail: anbumani_k@yahoo.co.uk

Keywords: digital watermarking, independent component analysis, discrete cosine transform

Received: September 19, 2007

This paper proposes a novel approach to content-based watermarking for image authentication that is based on Independent Component Analysis (ICA). In the scheme proposed here, ICA is applied to blocks of the host image and the resulting mixing matrix represents the features of the image blocks. Frobenius norm of the mixing matrix is adopted as the content-based feature. This is embedded as the watermark in a mid-frequency DCT coefficient of the block. This authentication technique is robust against incidental image processing operations, but detects malicious tampering and correctly locates the tampered regions.

Povzetek: Predlagana je nova metoda avtentikacije slik.

1 Introduction

A **digital watermark** is a piece of information that is hidden in a multimedia content, in such a way that it is imperceptible to a human observer, but easily detected by a computer. The principal advantage is that the watermark is inseparable from the content [1]. Digital watermarking is the process of hiding the watermark imperceptibly in the content. This technique was initially used in paper and currency as a measure of authenticity.

The primary tool available for data protection is encryption. Encryption protects content during the transmission of the data from the sender to receiver. However, after receipt and subsequent decryption, the data is no longer protected. Watermarking complements encryption [1].

Digital Watermarking involves two major phases:

(i) Watermark embedding, and
(ii) Watermark extraction.

Digital watermarks can be a pseudo random sequence or a logo of a company or an image. Watermark embedding is done in the watermark carriers such as Discrete Cosine Transform (DCT) or Discrete Wavelet Transform (DWT), etc of the original data resulting in watermarked data. The watermarked data may be compressed to reduce its size, corrupted by noise during its transmission through a noisy channel. It may be subjected to other normal image processing operations such as filtering, histogram modification etc. Also malicious intruders may tamper the data.

DCT is a widely used technique for watermarking [1]. Recently ICA is being adopted for watermarking [2] – [8]. In [2] ICA is applied to the blocks of the host image and the watermark image. The least-energy independent components of the host are replaced by the high-energy independent components of the watermark image. For watermark extraction the demixing matrices of both the watermark and the host images are required.

Dan Yu et al. [3] treat the host image, the key image, and the watermark image as the independent sources. Embedding is done by weighted addition of the key and the watermark to the host. For watermark extraction, two more mixtures are obtained by adding the key and the watermark using different weights. ICA is then applied to these mixtures to separate the host, the key, and the watermark. The host and the key are required for watermark extraction. In [4] the same procedure as in [3] has been used. The only difference is in the algorithm used for ICA.

Ju Liu et al. [5] use ICA for detection of the watermark which is a random sequence embedded in low-frequency DCT coefficients. Original DCT coefficients are required for watermark detection and for creating a second mixture needed for ICA.

Bounkong et al. [6] apply ICA to each block of the host image and obtain its independent components. The watermark is embedded in selected components using quantization and a modified image block is obtained from these modified independent components. This is added to the host image block, obtaining the watermarked image. In the extraction phase, ICA is applied to each block obtaining the independent components. The watermark is then extracted from these through dequantization.

The technique of creating three mixtures is also employed in [7] and [8]. While [7] uses upsizing and downsizing, [8] uses the so-called redundant DWT (RDWT).

Many authors have worked on content-based watermarking for image authentication. Li et al. [14] discuss a content-based watermarking scheme that uses local features of the image such as edges and zero-crossings. Their scheme uses a look-up table to embed the watermark and the same table is required at the receiver end to extract the watermark.

Content-based watermark is generated based on salient features of the image either in spatial domain like edges, texture, fractal dimensions [15] etc. or in a transform domain such as singular values [16], eigenvalues [17], etc. Choices of image features vary with techniques and directly influence the robustness of the scheme. Some techniques generate a random binary sequence to embed the watermark based on the features of the images [18] and [19]. A content-based digital signature scheme for image authentication has been presented in [20].

In [21] - [24] a localization based method has been presented to verify the integrity of the received image. In these techniques the host image is divided into a number of disjoint blocks and watermark is embedded in each of these blocks. To verify the authenticity of the received image, blockwise authentication has been done.

In [18], [25], [26] image authentication has been done using content-based watermarks. But these schemes do not embed the watermark in the image content; instead embed them in the image header. These techniques distort the host image prior to watermark embedding.

In [27] a watermarking technique based on the quadtree is proposed. This scheme embeds a Gaussian sequence watermark into low-frequency band of the wavelet transform. In their technique, watermark is embedded into visually insensitive pixels in quadtrees.

Most of the above authors embed a specified watermark. Most have copyright protection as their goal and require a lot of information about the host image for watermark extraction. Additional image mixtures are artificially created and then ICA is used as a blind source separation technique to separate the host image, the watermark, and the key.

In the scheme proposed here, a different approach to the use of ICA is adopted. ICA is used to determine the mixing matrix which – specifically its Frobenius norm – represents the content of the host image. No information about the host is required for watermark extraction. Thus the proposed scheme is a novel, blind, content-based watermarking for content authentication that uses ICA and DCT.

2 Proposed content-based watermarking using ICA

2.1 Principles of ICA

Principles of ICA are discussed in [9]–[13]. Let x be a random vector of observations. ICA models the observed data as: x = As, where the vector s represents the independent sources that generate the observed data x. The matrix A is the mixing matrix. Each x_j is a linear combination of the independent source signals. Hence xi is called the mixed signal. ICA estimates A and s from given x.

In the context of images, each row is considered as one observation x. Thus the entire image is denoted as X, representing all the rows of the image. ICA models an image X as: X = AS.

In the general case, X is m x n and S is r x n, where m $\geq$ r. In other words, the number of observed mixed signals must be greater than or equal to the number of independent components. Some significant points of ICA are:

- Each component s_j is independent of every other component.
- All independent components are non-Gaussian (with possible exception of one).
- The observation x is assumed centered.
- There is no order specified among the independent components.
- The independent components can be obtained from the observations as, s=Wx, where W is called the demixing matrix, estimated in ICA.
- Each column of the mixing matrix A represents a feature of the data x.

The last mentioned property is made use of in this proposed technique using ICA. Each column of the mixing matrix A represents a feature of the image. So, all the columns together represent all the features of the image. In order to get one single quantity to represent the image, the Frobenius norm of A has been chosen as the content-based feature that represents the image. The Frobenius norm of a matrix is the square root of the sum of the norms of all the columns of A.

The proposed content-based watermarking for blind authentication uses a hybrid of ICA and DCT. The host image is divided into small blocks. ICA is applied to each block and the mixing matrix of the block is determined. Frobenius norm of the mixing matrix is computed. This is considered as the content-based feature of the block. Such features are obtained for all the blocks of the host image. These constitute the content-based watermark used for authentication of the image.

In this technique, the watermark is embedded by replacing the chosen mid-frequency coefficient DCT(p, q) with a scaled value of the watermark but retaining the sign of the DCT coefficient, i.e.,

$$DCT(p,q) = sign(DCT(p,q)) * (\alpha * w)$$

where w is the content-based watermark. Here the watermark is the Frobenius norm of a block, which is always non-negative. The value for α is chosen based on the statistical details of the DCT coefficients and the watermark.

Watermark extraction is the reverse of the embedding process. The received watermarked image is divided into blocks and ICA is applied to each of them. The Frobenius norm of the mixing matrix is computed. DCT of each block is also performed. The watermark that was embedded is extracted from the chosen mid-frequency coefficient:

$$ExtractedWatermark = \frac{|DCT(p,q)|}{\alpha}$$

The percentage difference (Δ) between the extracted and the embedded watermarks is computed. If it is high, it indicates that the image has been tampered. Details of steps for the three phases of watermarking are given below

i. Watermark generation,
ii. Watermark embedding, and
iii. Watermark extraction and authentication.

Watermark generation

1. Segment the host image I of size n x n into blocks of size m x m resulting in K blocks.
2. Perform ICA of each block treating each row of the block as a vector.
3. Extract the mixing matrix A.
4. Compute the Frobenius norm of the mixing matrix; this is the content-based watermark w of the block.
5. Repeat steps 2 – 4 for computing the watermark for all the blocks. This set forms the watermark, $W = \{w_1, w_2, ..., w_k\}$

Watermark embedding

1. Perform DCT of each block.
2. Select the mid-frequency coefficient at the chosen location (p, q) in each block.
3. Replace the chosen coefficient with the watermark:

$$DCT\ (p,q) = sign(DCT\ (p,q)) * (\alpha * w)$$

4. Perform inverse DCT.
5. Repeat steps 1– 4 for all the blocks.

The resultant is the watermarked image I*.

Watermark extraction and authentication

1. Perform steps 1–5 of the watermark generation procedure on the received image I' and obtain the computed watermark, .
2. Perform DCT of each block.
3. Extract the embedded watermark from the chosen DCT coefficient:

$$w' = \frac{|DCT(p,q)|}{\alpha}$$

4. This set forms the extracted watermark,

$$W' = \{w'_1, w'_2, ..., w'_K\}$$

5. Calculate the blockwise percentage difference (Δ) between the watermark values w* and w':

$$\Delta = \frac{|w_i^* - w_i|}{\max\{w_i\}} * 100$$

In this scheme the percentage difference of the values corresponding to each block is used to detect any change in the block and thereby the authenticity of the image. If the difference is small – smaller than an experimentally chosen threshold value – the block and therefore the entire image is deemed authentic. If the difference of any block is greater than the threshold, that block is identified as the tampered block and hence the image is unauthentic. .

3 Experimental results

The proposed blind content-based watermarking scheme for image authentication has been tested using Matlab and Adobe Photoshop. The scheme has been evaluated on a set of three different categories of 512 x 512 gray scale images: (i) standard images, (ii) natural images, and (iii) images created using imaging tools.

Choice of parameters

In order to determine the block size for image segmentation various block sizes were tried. In [6] Bounkong has mentioned that choosing a block size is based on the processing time and relevant features. Blocks of small size leads to poor performance in watermarking process and larger blocks demand high computational time. Hence a trade off between these two is required to choose the block size. After experimentation, a block size of 16 x 16 was chosen as it resulted in better PSNR value, computational time and better feature representation.

In order to embed the watermark in a suitable location the proposed technique uses one of the mid-frequency coefficients. Embedding the watermark in low-frequency components, results in visual degradation of the host image. Similarly embedding the watermark data in high-frequency components is not advisable as they may be lost during compression. Hence embedding the watermark in mid-frequency components ensures robustness.

The mid-frequency coefficient (p, q) in which to embed the watermark is chosen as the mid-diagonal coefficient i.e. the location

$$\left(\frac{blocksize}{2}, \frac{blocksize}{2}\right)$$

For choosing a suitable value for the embedding strength α, statistics of the DCT coefficient values at that mid-diagonal location of all the blocks are obtained, specifically the standard deviation α_x. Similarly the standard deviation α_w is obtained for the watermark.

The value of embedding factor α is determined such that the watermark values are suitably scaled to have the same range of variation as that of the DCT coefficients: $\alpha = \frac{\alpha_x}{\alpha_w}$. In this experimentation after computation the value is α = 0.14.

Threshold for the percentage difference Δ between the watermarks has been experimentally determined as 15%. Lower thresholds resulted in false negatives; while higher thresholds made the technique to be fragile.

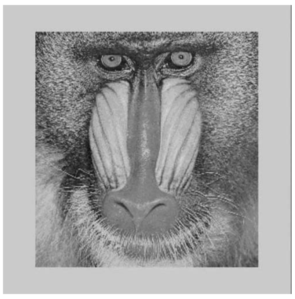
The ICA algorithm adopted in this proposed technique is the fastICA algorithm. This algorithm has been discussed in [9] and [13].

Quality of the watermarked image

The proposed content-based watermarking scheme has been implemented on a set of images of three categories. The metrics PSNR, Pearson Correlation Coefficient (PCC), Normalized Cross Correlation (NCC), and Image Fidelity (IF) are calculated between the host image and the watermarked image.

The test images after watermarking is shown in Fig. 1. It can be observed that there is no perceptually noticeable difference in the images due to watermarking.

Numerical values of the performance metrics for the test images given in Table 1 also corroborate this. PSNR values range from 96.09 to 102.45, with an average of 97.87 for all the test images**,** which is quite high. The other metrics, Pearson Correlation Coefficient (PCC) and Image Fidelity (IF) are also quite high. This shows that watermark embedding does not degrade the visual quality of the image.



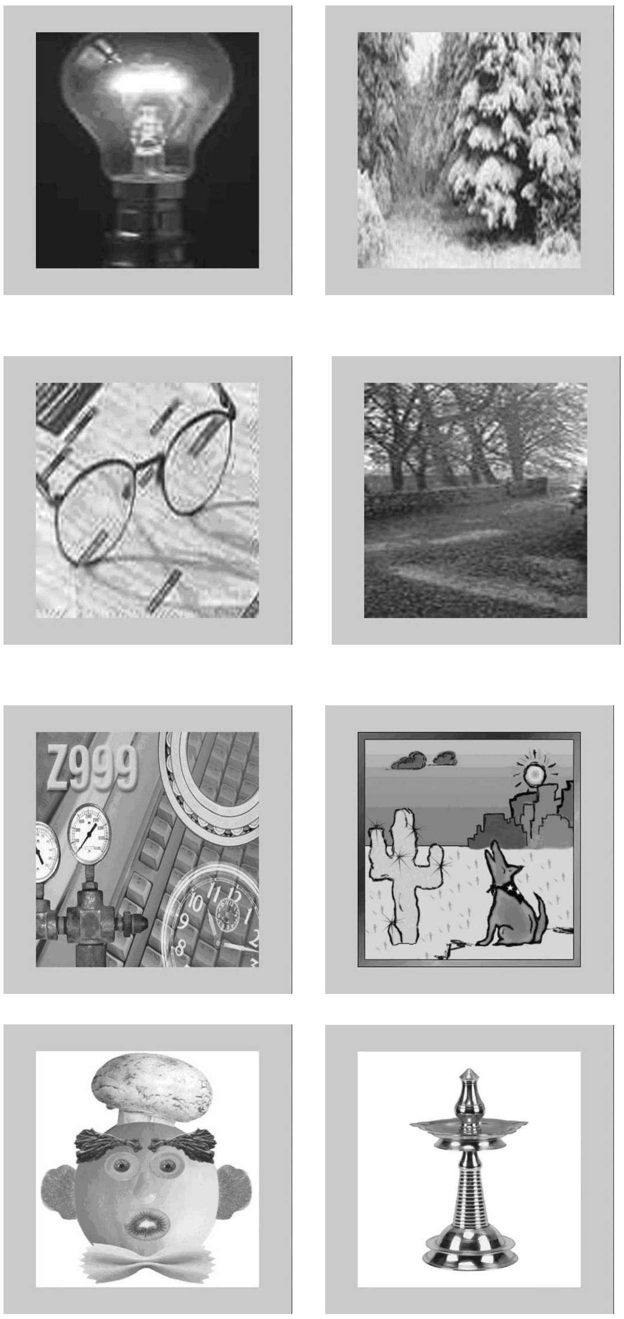


Figure 1: Images after watermarking using the ICA technique.

Extraction efficacy
The efficiency of the scheme in correctly extracting the watermark is given by the percentage difference between the computed and extracted Frobenius norm of the mixing matrix of the received image blocks. Table 2 gives the highest percentage difference Δ for some of the test images. The values are small, ranging from 1.87 to 9.60%, over all the test images**.** This indicates that the scheme extracts the embedded watermark accurately.

Table 1: Quality metrics after watermarking using ICA

	PSNR	IF	NCC	PCC
Bridge	98.8892	1.0000	1.0000	0.9999
Boy	98.6417	0.9999	1.0000	0.9999
Building	96.7310	0.9999	1.0000	0.9999
Cameraman	97.4249	1.0000	1.0000	0.9999
Clown	98.2414	0.9999	1.0000	0.9999
Couple	96.7900	0.9999	1.0000	0.9999
Jet Plane	98.6468	1.0000	1.0000	0.9999
Lena	98.0096	0.9999	1.0000	0.9999
Living Room	98.7276	1.0000	1.0000	0.9999
Mandrill	98.4030	1.0000	1.0000	0.9998
Peppers	96.0968	0.9999	1.0000	0.9998
Sail Boat	96.4620	1.0000	1.0000	0.9999
Bulb	102.4517	1.0000	1.0000	1.0000
Snow Tree	97.8179	1.0000	1.0000	0.9999
Specs	97.8651	1.0000	1.0000	0.9998
Trees	97.3633	0.9999	0.9999	0.9997
KeyClock	96.2998	1.0000	1.0000	0.9998
SunBark	96.8414	1.0000	1.0000	0.9999
Decor	97.2312	0.9999	0.9999	0.9989
Lamp	96.1245	0.9989	0.9999	0.9978
Average	97.7529	0.9999	1.0000	0.9997
Minimum	96.0968	0.9989	0.9999	0.9978
Maximum	102.4517	1.0000	1.0000	1.0000

Robustness against incidental image processing
Robustness of the proposed scheme against normal signal processing operations such as compression, noise and filtering has been experimentally evaluated on all the test images.

In this proposed watermarking technique the watermarked image is subjected to three types of distortions: compression, noise, and filter. Watermarked image has been compressed using JPEG compression with different quality factors. Additive white Gaussian noise (AWGN) and uniform noise has been added to the watermarked image. Also filtering such as low pass, sharpening, histogram equalization, and contrast stretching has been applied on the watermarked image. Results of the test image Lena is shown in Table 3.

For all these attacks, the values of highest percentage difference Δ, ranges from 1.25 to 6.45 with an average of 3.23 as given in Table 3. All the percentage differences all less than the determined threshold 15%, indicating that there is no tampering.

Table 2: Results after watermark extraction without attacks using ICA

Image	Highest percentage difference
Bridge	9.1100
Boy	8.3440
Building	8.5009
Cameraman	4.5248
Clown	8.6273
Couple	8.2430
Jet Plane	7.5966
Lena	8.9354
Living Room	9.2842
Mandrill	6.6092
Peppers	7.9915
Sail Boat	8.7662
Bulb	1.9682
Snow Tree	8.7526
Specs	8.2906
Trees	1.8791
KeyClock	9.6035
SunBark	8.2667
Decor	7.2345
Lamp	7.1124
Average	7.4820
Minimum	1.8791
Maximum	9.6035

Table 3: Results after incidental distortions on Lena using ICA

Attacks	Parameters	Highest percentage difference
JPEG Compression		
Maximum	Quality Factor =10	3.5728
High	Quality Factor =8	2.1245
Medium	Quality Factor =5	2.0000
Low	Quality Factor =3	1.9925
Noise		
AWGN	Percent = 5	6.4512
Uniform	Percent = 5	3.1928
Filter		
Low pass	Standard Deviation = 10	4.2686
Sharpening	-	3.2578
Histogram Equalization	-	4.2564
Gamma Correction	Gamma value = 3	3.2578
Contrast Stretching	Brightness = 15 Contrast = 15	1.2456
Average		3.2382
Maximum		6.4512
Minimum		1.2456

Similar good performance of robustness of the proposed scheme has been obtained for other test images also. For example, Fig. 2 shows the robustness of the various test images against:

- JPEG compression medium quality (Quality factor = 5)
- AWGN with noise 5%
- Low pass filter with standard deviation 10.

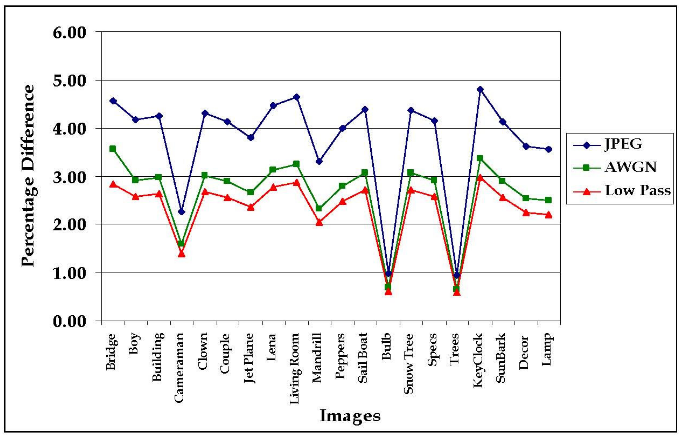


Figure 2: Robustness of ICA-based technique after incidental image processing

Detection of tampering

To demonstrate the ability of the proposed scheme in locating the blocks that have been intentionally tampered, the watermarked Lena image has been intentionally tampered by introducing small patches. In one case, a text has been placed at the top left corner of Lena, Fig. 3 (a), and the method correctly located the tampered blocks as shown in Fig. 3 (b). Patches were introduced at other points on Lena. In all the cases this technique correctly identified malicious tampering.

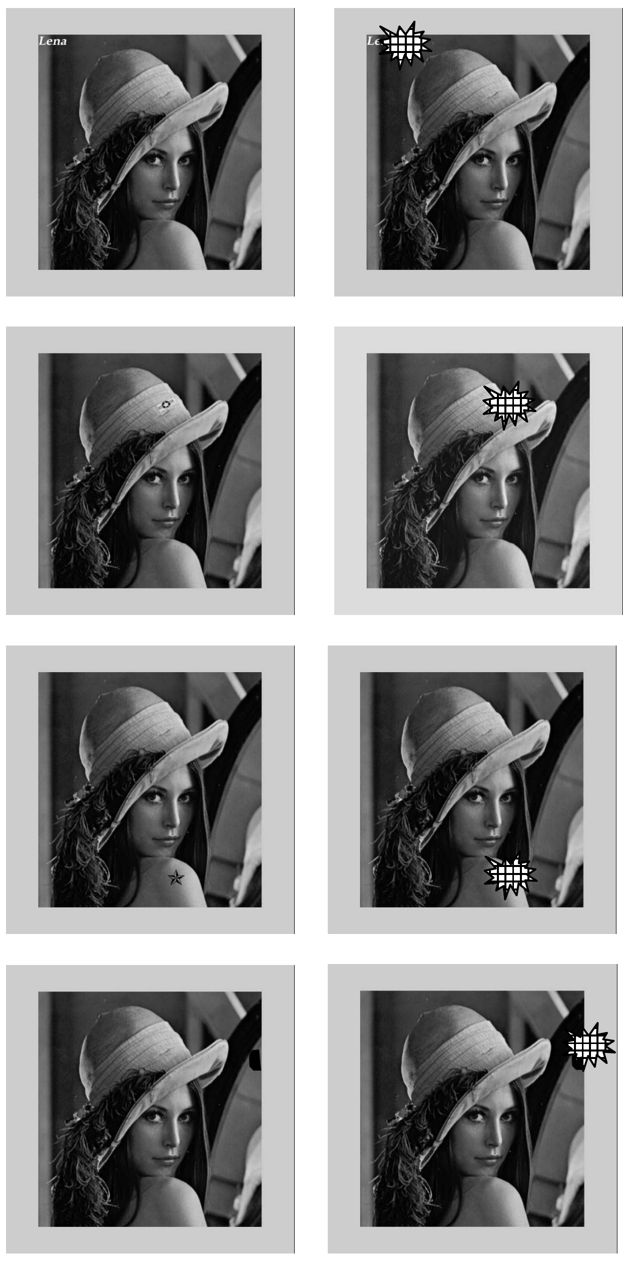


Figure 3: Lena with location of tampered blocks.

Table 4: Results after intentional tamper of Lena using ICA

Locations of tamper	Blocks identified as tampered	Percentage difference of tampered blocks
Top	(1,1) and (1, 2)	55 and36
Hat	(3, 18), (3, 19), and (3, 20)	27, 30, and 29
Shoulder	(30,51), (30, 52), (31,53), (31,51), and (31,52)	41, 61, 36, 35, and 31
Opposite to hat	(4,63), (4,64) (5,63), and (5,64)	31, 69, 32, and 65

4. Conclusion

This paper has discussed a new blind content-based watermarking scheme for image authentication using ICA and DCT. The watermark to be embedded is obtained from the host image itself in terms of the Frobenius norm of the mixing matrix obtained during ICA. These are embedded in the mid-frequency DCT coefficients. The proposed method correctly authenticates the image even under normal image processing operations and it correctly detects tampering and identifies the tampered regions of the image. The major quality of the watermarked image, this proposed technique is much superior to the techniques in the literature. The average PSNR value in the existing techniques is around 52.45; whereas the average PSNR value in this proposed ICA based technique is around 97. In terms of computation time, the ICA based technique takes a longer time. This is mainly due to the algorithms used to compute independent components. Exhaustive experimentation demonstrates the efficacy of the proposed scheme.

References

[1] Ingermar J. Cox, Matthew L. Miller, and Jeffrey A. Bloom, Digital Watermarking, Morgan Kaufmann Publishers, 2002.

[2] Francisco J. Gonzalez-Serrano, Harold. Y. Molina-Bulla, and Juan J. Murillo- Fuentes," Independent component analysis applied to digital image watermarking," International Conference on Acoustic, Speech and Signal Processing (ICASSP), vol. 3, pp. 1997-2000, May 2001.

[3] Dan Yu, Farook Sattar, and Kai-Kuang Ma, "Watermark detection and extraction using independent component analysis method," EURASIP Journal on Applied Signal Processing, vol. 1, pp. 92–104, 2002.

[4] Minfen Shen, Xinjung Zhang, and Lisha Sun, P. J. Beadle, F. H. Y. Chan, "A method for digital image watermarking using ICA," 4th International Symposium on Independent Component Analysis and Blind Signal Separation (ICA 2003), Nara, Japan, April 2003, pp. 209-214.

[5] Ju Liu , Xingang Zhang, Jiande Sun, and Miguel Angel Lagunas, "A digital watermarking scheme based on ICA detection," 4th International Symposium on Independent Component Analysis and Blind Signal Separation, (ICA 2003), Nara, Japan, April 2003, pp. 215-220.

[6] Stephane Bounkong, Boremi Toch, David Saad, and David Lowe, "ICA for watermarking digital images," Journal of Machine Learning Research 4, pp. 1471-1498, 2003.

[7] Viet Thang Nguyen and Jagdish Chandra Patra, "Digital image watermarking using independent component analysis," PCM 2004, Lecture Notes in Computer Science 3333, pp. 364-371, Springer-Verlag, 2004.

[8] Thai Duy Hien, Zensho Nakao, and Yen-Wei Chen, "Robust multi-logo watermarking by RDWT and ICA", Signal Processing, Elsevier, vol. 86, pp. 2981-2993, 2006.

[9] Aapo Hyvarinen, "Survey on Independent Component Analysis", Neural Computing Surveys, vol. 2, pp. 94-128, 1999.

[10] Hyvarinen, Karhunen, and Oja, "Introduction," Chapter 1 in Independent Component Analysis, John Wiley, pp. 1-12, 2001.

[11] Errki Oja, "Independent Component Analysis: Introduction," European Meeting on ICA, Vietri sul Mare, Feb. 21, 2002.

[12] Lucas Parra, "Tutorial on blind source separation and independent component analysis," Adaptive Image and Signal Processing Group, Sarnoff Corporation, Feb. 9, 2002.

[13] Bogdan Matei, "A review of independent component analysis technique," Tutorial, Electrical and Computer Engineering Department, Rutgers University, Piscataway, NJ, USA.

[14] Chang-Tsun Li, Der-Chyuan Lou, and Tsung-Hsu Chen, "Image authentication and integrity verification via content-based watermarks and a public key cryptosystem," Proceedings of International Conference on Image Processing, vol. 3, 2000 pp. 694 - 697

[15] Rongrong Ni, Quiqi Ruan, and H.D. Cheng, "Secure semi-blind watermarking based on iteration mapping and image features," Pattern Recognition, vol. 38, pp. 357-368, 2005.

[16] Herve Abdi, "Singular value decomposition and generalized singular value decomposition," in Neil Salkind (Ed.), Encyclopedia of Measurement and Statistics, 2007.

[17] Arto Kaarna, Pekka Toivnen, and Kimmo Mikkonen, "Watermarking spectral images through PCA transform," Proceedings of PICS, The Digital Prography Conference, May 2003, pp. 220-225.

[18] Chai Wah Wu, "On the design of content-based multimedia authentication systems," IEEE Transactions on Multimedia, vol. 4, no.3, pp. 385-393, September 2002.

[19] Eugene T. Lina, Christine I. Podilchuk, and Edward J. Delp, "Detection of image alterations using semi-fragile watermarks," Proceedings of SPIE International Conference on Security and watermarking of Multimedia contents, January 2000.

[20] Marc Schneider and Shih-Fu Chang, "A robust content based digital signature for image authentication," Proceedings of International Conference on Image Processing, vol. 3, September 1996, pp. 227-230.

[21] Roberto Caldelli, Franco Bartiloni, and Vito Cappellini, "Standard metadata embedding in a digital image," Proceedings of 14th International Workshop on Database and Expert Systems Applications, 2003.

[22] M.G. Albanesi, M. Ferretti, and F. Guerrini, "A taxonomy for image authentication techniques and its application to the current state of the art," Proceedings of the 11th IEEE International Conference on Image Analysis and Processing (ICIAP '01), 2001.

[23] Phen-Lan Lin, Po-Whei Huang, and An-Wei Peng, "A fragile watermarking scheme for image authentication with localization and recovery," Proceedings of the IEEE Sixth International Symposium on Multimedia Software Engineering (ISMSE'04), 2004.

[24] Huijuan Yang and Alex C. Kot, "Binary image authentication with tampering localization by embedding cryptographic signature and block identifier," IEEE Signal Processing Letters, vol. 13, no. 12, pp. 741-744, December 2006.

[25] Nasir Memon, Poorvi Vora, Boon-Lock Yeo, and Minerva Yeung, "Distortion bounded authentication techniques," Proceedings of International Conference on Watermarking and Multimedia Contents, February 2000.

[26] Chai Wah Wu, "Limitations and requirements of content-based multimedia authentication systems," Proceedings of International Conference of SPIE, vol. 4314, 2001, pp. 241-252.

[27] Kil-Sang Yoo, Mi-Ae Kim, and Won-Hyung Lee, "A robust image watermarking technique for JPEG images using quadtrees," Lecture Notes in Computer Science, vol. 3332, pp. 34-41, 2004.

A Distributed Multilevel Ant Colonies Approach

Katerina Taškova, Peter Korošec and Jurij Šilc
Computer Systems Department, Jožef Stefan Institute, Ljubljana, Slovenia
E-mail: katerina.taskova@ijs.si

Keywords: ant-colony optimization, distributed computing, mesh partitioning, multilevel approach

Received: May 20, 2008

The paper presents a distributed implementations of an ant colony optimization metaheuristic for the solution of a mesh partitioning problem. The usefulness and efficiency of the algorithm, in its sequential form, to solve that particular optimization problem has already been shown in previous work. In this paper a straightforward implementations on a distributed architecture is presented and the main algorithmic issues that had to be addressed are discussed. Algorithms are evaluated on a set of well known graph-partitioning problems from the Graph Collection Web page.

Povzetek: V sestavku je predstavljena porazdeljena izvedba metahevristične optimizacije s kolonijami mravelj, ki je uporabljena pri reševanju problema razdelitve mreže.

1 Introduction

Real engineering problems have complex dynamic behavior and one of the widely accepted formalisms for their modeling are partial differential equations (PDEs). The fraction of PDEs that have solutions in a closed analytical form is quite small and in general their solution relies on numerical approximations. Finite-element method is a well known numerical method that efficiently solves complex PDEs problems. In order to find an approximation of an unknown solution function $f(x)$, this method discretizes the underlying domain into a set of geometrical elements consisting of nodes. This process is known as meshing. The value of the function $f(x)$ is then computed for each of these nodes, and the solutions for the other data points are interpolated from these values [4].

Generated mesh structures can have large number of elements, therefore a common approach would involve a mesh-partitioning task in order to solve the finite-element method using multiple parallel processors. Consequently, the mesh-partitioning task aims to achieve minimal inter-processor communication and at the same time to maintain a processor workload balance.

Mesh-partitioning problem is a combinatorial optimization problem. Namely, it is a special case of the well-known graph-partitioning problem, which is known to be a NP-hard and is defined as follows: If $G(V,E)$ denotes an undirected graph consisting of a non-empty set V of vertices and a set $E \subseteq V \times V$ of edges, then k-partition D of G comprises k mutually disjointed subsets $D_1, D_2, \ldots, D_k$ (domains) of V whose union is V. The set of edges that connect the different domains of a partition D is called an edge-cut. A partition D is balanced if the sizes of the domains are roughly the same, i.e., if $b(D) = \max_{1\leq i\leq k} |D_i| - \min_{1\leq i\leq k} |D_i| \approx 0$. The graph-partitioning problem is to find a balanced partition with a minimum edge-cut, denoted by $\zeta(D)$.

Employing metaheuristic approach in optimization has introduced efficient and practical solution of many complex real-world problems. A variety of heuristic based methods are used for solving the mesh-partitioning problem as well [1, 10, 12]. In spite of being very powerful approach, metaheuristic can still easily reach the computational time limits for large and difficult problems. Moreover, heuristics do not guarantee an optimal solution, and in general their performance could depend on the particular problem setting. An important issue that arises here is not only how to design/calibrate the algorithm for a maximum performance, but also how to make it robust in terms of dealing with different types of problems and settings. Parallel processing is an straightforward approach that addresses both issues, computational time and robustness.

One relatively new and promising metaheuristic that is competitive with standard mesh-partitioning tools, such as Chaco [9], JOSTLE (that has recently been commercialised and is available under the name of NetWorks), and k-METIS [11], is known as *Multilevel Ant-Colony Algorithm* (MACA) [14]. This method is a nature inspired heuristic that uses population of agents (artificial ants) mediated by pheromone trails to find a desired goal, i.e., an ant-colony optimization algorithm [6] for solving mesh-partitioning problem. In experimental analysis so far, MACA has performed very well on different size test graph problems [14]. Since it is a population-based algorithm, MACA is inherently suitable for parallel processing on many levels. Motivated by the good performance of MACA in the previous work and the possibility to improve it's performance (computational cost and/or solution quality), in this paper we discus the result of parallelizing MACA on largest scale, executing entire algorithm runs concurrently on a multiple instruction stream, multiple data stream (MIMD) machine architecture. Explicitly, we present and experimentally evaluate two distributed versions of MACA, the Semi-Independent Distributed MACA and the Interactive Distributed MACA approach on a set of well known graph-partitioning problems from the Graph Partitioning Archive [8]. Both distributed approaches show comparable or better (stable) quality performance. Semi-independent distributed approach can obtain same or better quality for less computational time, which is gain on both scales: quality and cost.

The rest of the paper is organized as follows: Section 2 describes the MACA algorithm for solving the mesh-partitioning problem. Section 3 outlines possible parallel strategies and in detail describes the two distributed implementations of MACA. The experimental results are presented and discussed in Section 4. Conclusions and possible directions for further work are given in Section 5.

2 The multilevel ant-colony algorithm

The MACA is an ant-colony algorithm [6] for k-way mesh (graph) partitioning enhanced with a multilevel technique [17] for global improvement of the partitioning method. The MACA is a recursive-like procedure that combines four basic methods: graph partitioning (*the basic ant-colony optimization metaheuristic*), graph contraction (*coarsening*), partitioned graph expansion (*refinement*) and *bucket sorting*.

2.1 The basic ant-colony algorithm

The main idea of the ant-colony algorithm for k-way partitioning [13] is very simple: We have k colonies of ants that are competing for food, which in this case represents the vertices of the graph. Final outcome of ants activities is stored food in their nests, i.e., they partition the mesh into k submeshes.

The outline of the core optimization procedure in the MACA pseudocode is given in Algorithm 1. The algorithm begins with a initialization procedure that performs a random mapping of the input graph onto a grid, which represents the place where ants can move, locates the nests position on the grid and places the ants initially in their nest locus. While gathering food, the artificial ants perform probabilistic movements on the grid in three possible directions (forward, left and right), based on the pheromone intensity. When an ant finds food, it picks it up if the quantity of the temporarily gathered food in its nest is below a specified limit (the capacity of storage is limited in order to maintain the appropriate balance between domains); otherwise, the ant moves in a randomly selected direction. The weight of the food is calculated from the number of the cut edges created by assigning the selected vertex to the partition associated with the nest of the current ant. If the food is too heavy for one ant to pick it up then an ant sends a help signal (within a radius of a few cells) to its neighbor coworkers to help it carrying the food to the nest locus. On the way back to the nest locus an ant deposits pheromone on the trail that it is making, so the other ants can follow its trail and gather more food from that, or a nearby, cell. When an ant reaches the nest locus, it drops the food in the first possible place around the nest (in a clockwise direction)and starts a new round of foraging.

Along with foraging food, ants can gather food from other nests as well. In this case when the food is too heavy to be picked up, the ant moves on instead of sending a help signal. In this way the temporary solution is significantly improved. Furthermore, the algorithm tries to maintain a high exploration level by restoring cells pheromone intensity to the initial value whenever the pheromone intensity of a certain cell drops below a fixed value.

2.2 The multilevel framework

The multilevel framework [2] as presented in Algorithm 2 and Fig. 3 combines a level based coarsening strategy together with a level based refinement method (in reverse order) to promote faster convergence of the optimization metaheuristic and solution to a larger problems.

Coarsening is a graph contraction procedure that is iterated L times (on L levels). Adequately, a coarser graph $G_{\ell+1}(V_{\ell+1}, E_{\ell+1})$ is obtained from a graph $G_\ell(V_\ell, E_\ell)$ by finding the largest independent subset of graph edges and then collapsing them. Each selected edge is collapsed and the vertices $u_1, u_2 \in V_\ell$ that are at either end of it are merged into the new vertex $v \in V_{\ell+1}$ with weight $|v| = |u_1| + |u_2|$. The edges that have not been collapsed are inherited by the new graph $G_{\ell+1}$ and the edges that have become duplicated are merged and their weight summed. Because of the inheritance the total weight of the graph remains the same and the total edge weight is reduced by an amount equal to the weight of the collapsed edges, which have no impact on the graph balance or the edge-cut.

Refinement is a graph expansion procedure that applies on a partitioned graph G_ℓ (partitioned with the ant-colony algorithm), which interpolates it onto its parent graph $G_{\ell-1}$. Because of the simplicity of the coarsening procedure, the interpolation itself is a trivial task. Namely, if a vertex $v \in V_\ell$ belongs to the domain D_i, then after the refinement the matched pair $u_1, u_2 \in V_{\ell-1}$ that represents the vertex v, will also be in the domain D_i. In this way we expand the graph to its original size, and on every level ℓ of our expansion we run our basic ant-colony algorithm.

Large graph problems and the multilevel process by itself induce rapid increase of the number of vertices in a single cell as the number of levels goes up. To overcome this problem MACA employs a method, based on the basic *bucket sort* idea [7], that accelerates and improves the algorithm's convergence by choosing the most "promising" vertex from a given cell. Inside the cell, all vertices with a particular gain g are put together in a "bucket" ranked g and

```
Procedure Ant_Colony_Algorithm
    For all ants of colony Do
        For all colonies Do
            If carrying food Then
                If in nest locus Then Drop_Food()
                Else Move_to_Nest()
                End If
            Else If food here Then Pick_Up_Food()
                Else If food ahead Then Move_Forward()
                    Else If in nest locus Then Move_To_Away_Pheromone()
                        Else If help signal Then Move_To_Help()
                            Else Follow_Strongest_Forward_Pheromone()
                            End If
                        End If
                    End If
                End If
            End If
        End For
    End For
End Ant_Colony_Algorithm.
```

Figure 1: Basic ant-colony algorithm

```
Procedure Multilevel_Framework
    structure[0] = Initialization()
    For $\ell = 0$ To $L - 1$ Do
        structure[$\ell + 1$] = Coarsening(structure[$\ell$])
    End For
    For $\ell = L$ Downto $0$ Do
        Solver(structure[$\ell$])
            If $\ell > 0$ Then
                structure[$\ell - 1$] = Refinement(structure[$\ell$])
            End If
    End For
End Multilevel_Framework.
```

Figure 2: Multilevel framework

all nonempty buckets, implemented as double-linked list of vertices, are organized in a 2-3 tree. Additionally, MACA keeps separate 2–3 tree for each colony on every grid cell that has vertices in order to gain even faster searches.

3 Distributed multilevel ant-colony approaches

In general, ant-colony optimization algorithms can be parallelized on four different levels [5, 15, 16], as follows: (i) *parallelism on colony level*, (ii) *parallelism on ant level*, (iii) *data level parallelization*, and (iv) *functional parallelization*, where each one is differing in granularity and communication overhead between processors. We will in brief, in the first subsection, describe all four parallelization approaches, making a ground base for introduction of the proposed Semi-Independent Distributed MACA and Interactive Distributed MACA approaches in the second, and the third subsection, respectively.

3.1 Parallelization strategies

(i) *Parallelism on colony level* is the most simple coarse-grained parallelization of the ant-colony optimization algorithms, where the problem is instantiated and solved simultaneously on all available processors. Furthermore, if no communication is required between processors (parallel independent algorithms searches, introduced by Stützle [16]), then this approach is refereed to as *parallel independent ant colonies* and it is suitable for algorithms that perform stochastic searches. Otherwise, if colonies, while searching for food, exchange information at a specified iteration (requires synchronized communication which implies master/slave implementation), then we refer to this approach as *parallel interactive ant colonies*. The communication cost of the second approach can become very expensive due to the required broadcasting of entire pheromone structures.

(ii) *Parallelism on ant level* is the first proposed parallel implementation [3] of an ant-colony optimization algorithm, where each ant (or a group of ants) is assigned a separate processor to build a solution. This means maintenance of a separate pheromone structures on every processor and therefore this approach requires a master processor that will synchronize the work of the rest (slave processors), including ant-processor scheduling, initializations, global pheromone updates and producing of the final solution.

(iii) *Data level parallelization* is a suitable approach for solving the multi-objective optimization problems, since it divides the main problem into a number of subproblems (objectives to optimize) and each one is solved by a colony on a separate processor.

(iv) *Functional parallelization* is a parallelization that introduces a concurrent execution of a specified operations (local search, solution construction, solution evaluation) performed by a single colony on a master-slave architecture. When local heuristic searches are computationally expensive, a so-called *parallel local searches* are the preferred case. In particular, the assignment of a slave processor is to refine the solutions received from the master with local search heuristics, while the master is responsible for a solution construction, pheromone updates and collection of the refined solutions. The *parallel solution construction* is a second approach that organizes the available slave processors in two operational groups. Processors in the first one are responsible for a solution construction, while the second group processors are additionally grouped and scheduled to refine the corresponding solutions constructed by the first group processors. The last functional parallelization approach is called *parallel evaluation of solution elements*. This approach gives best performance in case of a computationally expensive solution evaluations. Compared to all aforementioned parallel strategies parallel evaluation of solution elements is the only approach that does not exploits parallelism within the metaheuristic algorithm.

An efficient parallelization of a given algorithm depends mainly on the available computing platform, the underlying problem and the algorithm itself. If there is a large communication overhead between the processors, then parallel performance can be degraded. When the algorithms uses global structures, such as the pheromone matrix or the grid matrix of 2–3 trees in MACA case, a shared memory system would gain on communication (less) over a distributed memory system. On the other hand, the most common and cheaper approach in the same time is a parallelization using distributed memory systems, i.e., MIMD architecture such as cluster of workstations. Our proposed MACA parallelization assumes distributed memory system as well and it is implemented on a cluster of workstations.

3.2 The semi-independent distributed MACA

The Semi-Independent Distributed MACA (SIDMACA) is basically a distributed MACA approach that allows exchange of the best temporal solution at the end of every level of the multilevel optimization process. This exchange requires that the parallel executions of MACA instances on the available processors have to be synchronized once per level. Namely, the master processor is responsible for synchronizing the work of all slave processor that execute a copy of MACA, by managing the exchange information and communication process (sending commands and confirmation, such as Start, Stop, Initialize, Goto New Level, Best Partition, etc.), while the slave processors have to execute the instances of the MACA code, signal when finish the current level optimization and send the best partition to the master. When all finish the current level, the master determines the best solution and broadcasts it to the slaves. In order to proceed with next level optimization, slave processors have to first update local memory structures (grid matrix) and afterwards perform partition expansion (refine-

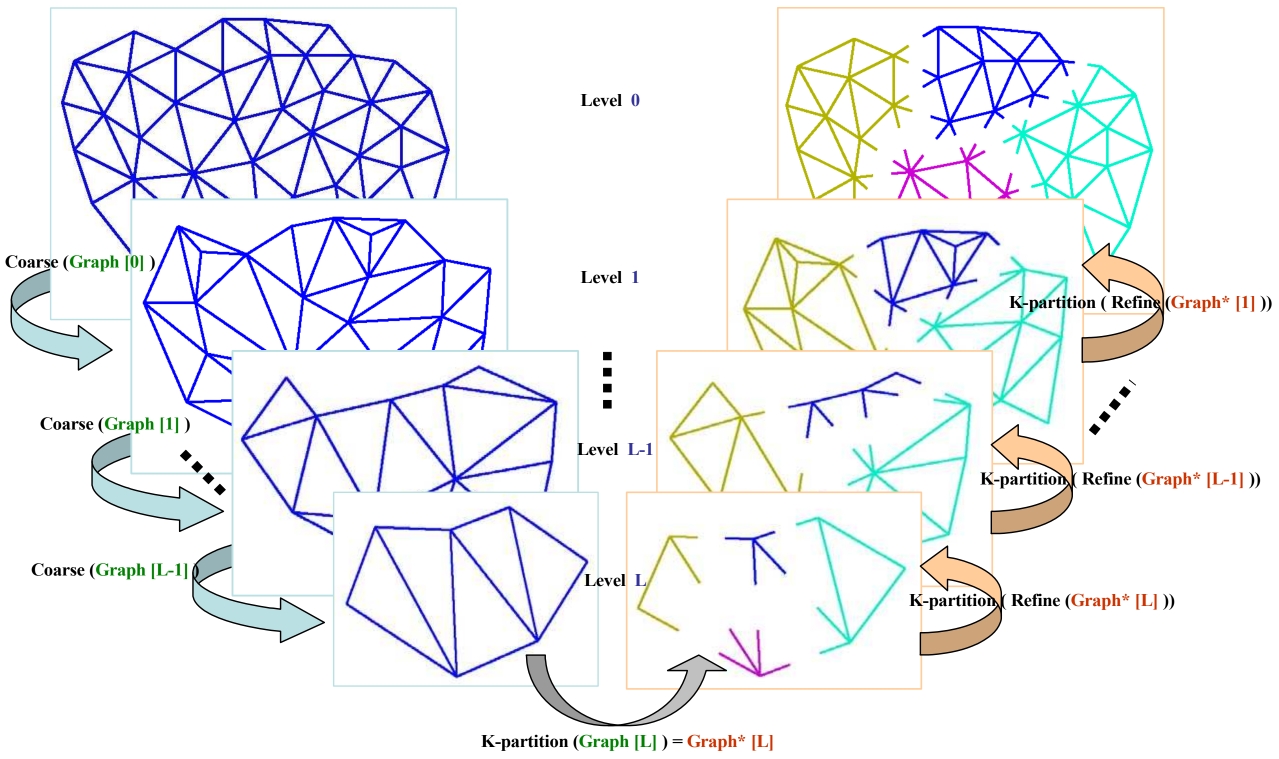

Level 0
Level 1
Level L-1
Level L
Coarse (Graph [0])
Coarse (Graph [1])
Coarse (Graph [L-1])
K-partition (Graph [L]) = Graph* [L]
K-partition (Refine (Graph* [L]))
K-partition (Refine (Graph* [L-1]))
K-partition (Refine (Graph* [1]))

Figure 3: The three phases of multilevel k-way graph partitioning.

ment).

3.3 The interactive distributed MACA

The Interactive Distributed MACA (ItDMACA) is based on the parallel interactive colony approach which, by definition, implies master/slave implementation and synchronized communication. The information exchange between the colonies across the concurrent processors is initiated every time a piece of food has been taken or dropped on a new position. The information about the specific food, its new position and its owner is part of the message sent to and received from the master processor when picked up or dropped food. The master keeps and updates its own local grid matrix of temporal food positions (plays the role of shared memory) in order to maintain normal and consistent slaves activities.

The master processors is responsible for the synchronized work and communication of the slave processors, which includes listening, processing and broadcasting of the incoming clients messages during level optimization. When all slave processors finish level or run, it collects the best-level solution, determines and broadcast the global best-level solution to the slaves and guides them when to start the refinement procedure and all necessary updates in order to perform the next level optimization activities or a new run.

A slave processor executes a single instance of the MACA code. While optimization executing informs the master and waits for master's confirmation on every potential drop/pick, signals when finishes the current level optimization and send the best partition to the master. In the meantime, while waiting to go on the next level, it listens for an eventual changes send by the unfinished clients and performs the eventual updates on the grid. When the master signals that the current level is finished, by sending the new best temporal solution, the slave processor has to perform partition expansion (refinement) in order to start the next level optimization.

4 Experimental evaluation

The proposed distributed versions of MACA were applied on a set of well-known graph problems and the results from their experimental evaluation are presented and discussed in this section. The section is structured in two subsection. The first subsection describes the implementation of the distributed code, the experimental setting and the benchmark suite, whereas the second subsection presents and discusses the evaluation results.

4.1 Setup

Based on the MACA sequential code, both proposed distributed version, SIDMACA and ItDMACA, were implemented in Borland® Delphi^{TM}, using TCP/IP protocol for the server/client communication, based on the open source library Indy Sockets 10 (which supports clients and servers of TCP, UDP and RAW sockets as well as over 100 higher level protocols).

All experiments were performed on a 8-node cluster connected via a Giga-bit switch, where each node consists of two AMD Opteron$^{\text{TM}}$1.8-GHz processors, 2GB of RAM, and Microsoft®Windows®XP operating system.

The benchmark graphs used in the experimental analysis were taken from the Graph Collection Web page [8], and are described in Table 1.

Graph $G(V,E)$	$\lvert V\rvert$	$\lvert E\rvert$
grid1	252	476
grid2	3296	6432
crack	10240	30380
U1000.05	1000	2394
U1000.10	1000	4696
U1000.20	1000	9339

Table 1: Benchmark graphs

	Number of processors				
Parameters	**1**	**2**	**4**	**6**	**8**
ants/colony	120	60	30	20	15
iteration/level	600	600	600	600	600
runs	20	20	20	20	20

Table 2: Distribution of ants per colonies and number of iterations per level w.r.t the number of processors

The total number of ants per colony was 120. As presented in Table 2, the number of ants per sub-colony is different and depends on the number p of processors, i.e., $\frac{1}{p}$ of the total number of ants, while the number of total iterations per level per colony is constant.

All experiments were run 20 times on each graph with each algorithm and as final results were presented the mean value (also best and worst values for edge-cut) of the considered evaluation criteria over all performed runs.

4.2 Results

The results presented in the following tables show the performance of the introduced DMACA approaches on the 2-partitioning and 4-partitioning graph problem. The *quality* of the partitioned graph is described with the edge-cut, $\zeta(D)$, and the balance, $b(D)$. Balance is defined as the difference (in the number of vertices) between the largest and the smallest domain.

Beside the quality, the second evaluation criteria is the effectiveness of the parallel algorithm which is, in our case, given by the *speed-up* measure, S, defined as:

$$S(p) = \frac{t_{\text{S}}}{t_{\text{T}}(p)}$$

and by the *relative speed-up* measure, S_r, which is defined as:

$$S_r(p) = \frac{t_{\text{T}}(1)}{t_{\text{T}}(p)},$$

where t_{S} is the time to solve a problem with the sequential code, $t_{\text{T}}(1)$ is time to solve a problem with the parallel code with the one processor, and $t_{\text{T}}(p)$ is time to solve the same problem with the parallel code on p processors. Note that $S(p)$ and $S_r(p)$ were calculated based on the average time values of the 20 runs.

By theory, correct speed-up metric should be calculated according to the performance (elapsed computational time) of the best serial code for the underlying algorithm, as defined above and denoted with $S(p)$, whereas in practice this is usually translated into calculation of the relative speed-up metric $S_r(p)$, since the best serial code is not available and writing two codes is not acceptable. In our case the serial code is available, and the values of both speed-up metrics are included in the tables with results.

Additionally, for the reason of comparison, in the tables are given the measured *CPU time* for the computation of the obtained solutions, t_{T}, as a triple of the time spent on pure computations, the time for communication with the master processor, t_{C}, and the time for internal updates caused by the synchronization, t_{U}. Note that t_{C} and t_{U} are part of the t_{T} spent for communication and updates, respectively.

Results in in Table 3 and Table 4 summarize the performance of SIDMACA for solving 2-partitioning and 4-partitioning graph problem, respectively, on the given graph set.

General observation is that parallel performance of the system w.r.t speed-up over the serial MACA is poor compared to the theoretical expected speed-up of p when used p processors, having maximal speed-up of 2.29 (graph ***crack***, $p = 8$) in case of 2-partitioning problem and maximal speed-up of 2.72 (graph ***U1000.05***, $p = 8$) in case of 4-partitioning problem overall considered graphs and parallel scenarios ($p = 2, 4, 6, 8$). For more then 2 processors employed $S > 1$ (except for the graph ***U1000.10***, $p = 4$, $k = 4$), while for 2-processor parallelization of the problems is evident speed-down up to 27% in case of 4-partitioning of graph ***grid2***. On the other side, results on SIDMACA show overall comparable/improved quality of the obtained solutions. The best solutions found in case of 2-partitioning are equal or better then the best serial code produced solutions (except for graph ***U1000.10***, $p = 4$ and ***crack***, $p = 6$). Moreover, the worst solutions found by SIDMACA are better than the ones from the MACA on the ***U1000*** graph set and ***crack*** graph. When solved the 4-partitioning problem, best found solution better than the best ones from the serial code are observed for graphs: ***grid2***, ***U1000.05*** and ***U1000.10***. The remark on the better quality of the worst case found solutions is confirmed in case of graphs ***U1000.10***, ***U1000.10*** and partially for graphs ***grid2***, ***U1000.05*** and ***crack***.

Correspondingly, Table 5 and Table 6 illustrate the ItD-MACA performance on the same graph set for the 2- and 4-partitioning graph problems when 2, 4 and 8 processor employed in parallel. Note that for $p = 8$ results are available only for the graphs ***grid1***, ***U1000.10*** and ***U1000.20***.

		Quality				**Time** [s]		**Speed-up**	
			$\zeta(D)$		$b(D)$	t_{T}	t_{C}	$S(p)$	$S_r(p)$
Graph	p	best	mean	worst	mean	mean	mean	mean	mean
	1*	18	18	18	0	9.80	0	1.00	
grid1	1	18	18	18	0	10.03	0.10		1.00
	2	18	18	18	0	10.41	0.69	0.94	0.96
	4	18	18	19	0	10.00	2.67	0.98	1.00
	6	18	18	21	0	7.17	1.75	1.37	1.40
	8	18	19	21	0	5.60	1.44	1.75	1.79
	1*	35	44	68	0	20.37	0	1.00	
grid2	1	34	41	68	0	23.81	0.21		1.00
	2	35	40	69	0	23.28	1.58	0.88	1.02
	4	35	40	69	0	15.86	2.99	1.28	1.50
	6	35	41	70	0	11.73	2.51	1.74	2.03
	8	35	49	70	0	9.31	2.22	2.19	2.56
	1*	1	1	2	0	87.53	0	1.00	
U1000.05	1	1	1	3	0	88.80	0.39		1.00
	2	1	1	2	0	83.10	1.81	1.05	1.07
	4	1	1	1	0	60.86	4.54	1.44	1.46
	6	1	1	1	0	42.15	6.09	2.08	2.11
	8	1	1	1	0	32.19	6.16	2.72	2.76
	1*	50	62	78	1	14.49	0	1.00	
U1000.10	1	39	62	73	1	15.03	0.17		1.00
	2	40	59	76	1	14.97	1.16	0.97	1.00
	4	40	59	71	1	11.88	2.57	1.22	1.27
	6	50	61	72	1	8.72	1.95	1.66	1.72
	8	57	61	72	1	7.12	1.76	2.04	2.11
	1*	221	277	370	8	12.14	0	1.00	
U1000.20	1	221	256	337	6	13.03	0.15		1.00
	2	219	259	373	7	12.48	0.99	0.97	1.04
	4	219	266	369	7	10.67	2.51	1.14	1.22
	6	219	288	368	10	7.75	1.75	1.58	1.68
	8	219	278	370	9	6.05	1.34	2.01	2.15
	1*	185	211	234	1	64.91	0	1.00	
crack	1	184	204	277	1	85.02	0.29		1.00
	2	184	195	231	0	80.48	6.28	0.81	1.06
	4	185	203	246	0	52.25	10.40	1.24	1.63
	6	186	202	230	0	39.70	9.25	1.64	2.14
	8	185	203	225	0	32.04	8.45	2.03	2.65

* sequential code

Table 3: Experimental results: 2-partitioning problem with SIDMACA

Graph	p	**Quality**				**Time** [s]		**Speed-up**	
		$\zeta(D)$			$b(D)$	t_{T}	t_{C}	$S(p)$	$S_r(p)$
		best	mean	worst	mean	mean	mean	mean	mean
grid1	1*	38	39	41	1	18.01	0	1.00	
	1	38	39	42	1	19.67	0.08		1.00
	2	38	39	41	0	19.38	0.67	0.93	1.01
	4	38	39	41	0	18.12	2.72	0.99	1.09
	6	38	39	41	0	13.50	1.69	1.33	1.46
	8	38	39	41	0	10.42	1.14	1.73	1.89
grid2	1*	96	104	118	4	47.38	0	1.00	
	1	95	102	111	3	63.08	0.23		1.00
	2	94	106	116	3	65.21	4.78	0.73	0.97
	4	92	105	123	2	47.96	10.14	0.99	1.32
	6	93	106	116	2	35.35	8.18	1.34	1.78
	8	93	103	115	2	28.16	6.89	1.68	2.24
U1000.05	1*	9	14	20	3	50.78	0	1.00	
	1	7	14	22	2	57.88	0.20		1.00
	2	9	14	23	2	60.96	8.28	0.83	0.95
	4	7	14	21	1	45.15	12.00	1.12	1.28
	6	8	13	18	0	36.26	9.99	1.40	1.60
	8	7	11	17	0	36.03	11.15	1.41	1.61
U1000.10	1*	95	114	166	3	35.17	0	1.00	
	1	102	113	133	3	43.09	0.12		1.00
	2	98	110	133	2	40.76	2.89	0.86	1.06
	4	92	112	163	2	39.03	10.12	0.90	1.10
	6	101	113	162	2	27.14	6.64	1.30	1.59
	8	91	115	161	3	19.96	4.64	1.76	2.16
U1000.20	1*	485	580	856	6	32.27	0	1.00	
	1	479	592	838	6	36.77	0.13		1.00
	2	485	586	817	6	36.19	1.85	0.89	1.02
	4	490	593	687	5	32.29	7.85	1.00	1.14
	6	490	632	730	6	22.65	4.70	1.42	1.62
	8	491	649	727	8	17.02	3.34	1.90	2.16
crack	1*	373	415	522	15	191.07	0	1.00	
	1	374	425	496	14	259.03	0.27		1.00
	2	377	426	495	11	217.40	14.39	0.88	1.19
	4	373	423	506	8	139.52	25.92	1.34	1.86
	6	384	431	493	6	109.29	23.66	1.75	2.37
	8	378	429	526	6	83.35	18.40	2.29	3.11

* sequential code

Table 4: Experimental results: 4-partitioning problem with SIDMACA

		Quality				**Time** [s]			**Speed-up**	
		$\zeta(D)$			$b(D)$	t_{T}	t_{C}	t_{U}	$S(p)$	$S_r(p)$
Graph	p	best	mean	worst	mean	mean	mean	mean	mean	mean
grid1	1*	18	18	18	0	9.80	0	0	1.00	
	1	18	18	18	0	44.79	34.45	0.11		1.00
	2	18	18	18	0	37.61	17.87	1.54	0.26	1.19
	4	18	18	18	0	18.86	8.26	3.01	0.52	2.38
	8	18	18	18	0	11.83	5.01	3.00	0.83	3.79
grid2	1*	35	44	68	0	20.37	0	0	1.00	
	1	35	45	69	0	143.34	118.24	0.29		1.00
	2	34	42	68	0	92.74	56.55	9.08	0.22	1.55
	4	35	39	53	0	52.29	26.57	13.44	0.39	2.74
U1000.05	1*	1	1	2	0	87.53	0	0	1.00	
	1	1	1	2	0	463.82	373.45	0.32		1.00
	2	1	1	2	0	295.38	190.25	41.64	0.30	1.57
	4	1	1	2	0	182.04	90.89	61.65	0.48	2.55
U1000.10	1*	50	62	78	1	14.49	0	0	1.00	
	1	39	60	76	1	44.47	27.11	0.20		1.00
	2	39	63	77	1	30.27	11.54	4.58	0.48	1.47
	4	40	59	71	1	21.16	6.63	4.77	0.68	2.10
	8	40	59	70	1	14.73	4.45	4.32	0.98	3.02
U1000.20	1*	221	277	370	8	12.14	0	0	1.00	
	1	219	268	373	7	24.41	11.23	0.15		1.00
	2	219	272	371	8	21.83	5.43	2.58	0.56	1.12
	4	219	255	368	7	16.48	2.77	3.92	0.74	1.48
	8	235	262	308	5	10.65	1.73	3.14	1.14	2.29
crack	1*	185	211	234	1	64.91	0	0	1.00	
	1	184	191	262	0	312.25	205.08	0.42		1.00
	2	184	189	211	0	223.60	95.82	57.03	0.29	1.40
	4	184	187	207	0	150.94	48.21	63.33	0.43	2.07

* sequential code

Table 5: Experimental results: 2-partitioning problem with ItDMACA

The results show no speed-up in case of 2-processor and 4-processor parallelization. Speed-up $S \geq 1$ is evident when 8 processor applied on the graphs for solving the 4-partitioning problem and for 2-partitioning of graphs ***U1000.10***, ***U1000.20***. Speed-down and low speed-ups are due to the big amount of time spent on communication and memory updates (synchronizations) during level optimization activities. The performance of ItDMACA w.r.t the quality of obtained solutions confirms the observation from the SIDMACA results. More specifically for the 2-partitioning problem, equal partition solutions in all runs are obtained for graphs ***grid1*** and ***U1000.05***, while significant improvement is evident for the ***U1000.10***, and slightly better solution for the rest of the graphs.

In general, comparable or improved solution quality is observed in the case of solving the 4-partitioning problem with ItDMACA as well. For $p = 8$, we gain speed-up and (i) better solution for graph ***U1000.20***, (ii) equal best found solution for graph ***grid1***, (iii) comparable solutions for graph ***U1000.10***.

As expected, the results on relative speed-up $S_r(p)$ are better than the speed-up $S(p)$ results. How big this difference is, is dependent on the size of the problem and algorithm implementation. Consequently, for SIDMACA the difference is not significant (except for graph ***grid2*** and ***crack***) compared to the ones in the ItDMACA, which in case of the ***grid2*** graph yields 7 times higher $S_r(p)$ than $S(p)$. This difference reveals that ItDMACA suffers from communication/update overhead, which for specific problems could be disadvantageous.

Additional experiments are needed in order to confirm the conclusions drawn from the initial experimental evaluations results, based on small number of processing nodes and a small set of graphs. There is a large space with possible directions for further work, such as:

– application on additional new graph problems, specially large and complex ones,

– try to solve the partitioning problem with more than 8 processors in parallel and find how the number of processors influences the solution quality and speed-up,

– shared memory implementation, since distributed implementations suffer from increased communication

Graph	p	**Quality**				**Time** [s]			**Speed-up**	
		$\zeta(D)$			$b(D)$	t_{T}	t_{C}	t_{U}	$S(p)$	$S_r(p)$
		best	mean	worst	mean	mean	mean	mean	mean	mean
grid1	1*	38	39	41	1	18.01	0	0	1.00	
	1	38	40	42	0	58.03	38.39	0.28		1.00
	2	38	40	43	0	47.97	16.25	1.38	0.38	1.21
	4	38	39	41	1	26.45	11.07	3.35	0.68	2.19
	8	38	40	43	1	14.00	5.43	2.22	1.29	4.15
grid2	1*	96	104	118	4	47.38	0	0	1.00	
	1	94	106	116	4	332.74	259.84	0.89		1.00
	2	95	103	114	4	210.78	110.65	45.41	0.22	1.58
	4	95	105	118	4	132.13	66.09	30.91	0.36	2.52
U1000.05	1*	9	14	20	3	50.78	0	0	1.00	
	1	9	14	21	3	342.12	278.76	0.62		1.00
	2	7	16	33	3	225.56	134.97	37.33	0.23	1.52
	4	7	15	22	2	160.29	91.36	38.15	0.32	2.13
U1000.10	1*	95	114	166	3	35.17	0	0	1.00	
	1	93	116	159	3	79.74	34.02	0.53		1.00
	2	96	112	129	3	63.46	17.23	7.32	0.55	1.26
	4	98	117	197	5	46.29	8.99	9.70	0.76	1.73
	8	98	118	157	3	26.08	5.13	7.34	1.35	3.06
U1000.20	1*	485	580	856	6	32.27	0	0	1.00	
	1	480	594	888	8	63.64	25.31	0.49		1.00
	2	487	583	759	6	51.82	11.07	4.39	0.62	1.22
	4	486	594	762	5	36.72	6.65	7.51	0.88	1.73
	8	474	584	805	5	24.25	3.52	6.50	1.33	2.62
crack	1*	373	415	522	15	191.07	0	0	1.00	
	1	372	415	507	15	720.23	401.47	1.11		1.00
	2	377	433	496	11	565.13	194.86	150.72	0.34	1.27
	4	382	415	492	9	411.00	104.70	197.98	0.46	1.75

* sequential code

Table 6: Experimental results: 4-partitioning problem with ItDMACA

and local memory updates,

– how statistically significant is the difference in the performances of the proposed parallel implementations among them or/and vs. the sequential MACA algorithm.

5 Conclusions

An efficient parallelization of a given algorithm depends mainly on the available computing platform, the underlying problem and the algorithm itself. If there is a large communication overhead between the processors, then parallel performance can be degraded. When the algorithms uses global structures, such as the pheromone matrix or the grid matrix of 2–3 trees in MACA case, a shared memory system would gain on communication (less) over a distributed memory system. On the other hand, the most common and cheaper approach in the same time is a parallelization using distributed memory systems, i.e., MIMD architecture such as cluster of workstations.

In this paper, two distributed MACA versions were presented, Semi-Independent and Interactive, implemented on a cluster of workstations. The initial experimental evaluations confirms that parallelization efficiency is problem dependent. Overall, both approaches show comparable or better (stable) quality performance. While ItDMACA is more sensitive on the parallel performance efficiency, due to the synchronization overhead, SIDMACA can obtain same or better quality for less computational time, which is gain on both scales: quality and cost.

In order to see how significant is this improvement and how robust is this approach additional experimental analysis regarding different problem type (large and complex) and experiment setup should be performed.

References

[1] A. Bahreininejad, B.H.V. Topping, and A.I. Khan. Finite Element Mesh Partitioning Using Neural Networks. *Adv. Eng. Softw.*, 27(1-2):103–115, 1996.

[2] S.T. Barnard and H.D. Simon. A Fast Multilevel Implementation of Recursive Spectral Bisection for

Partitioning Unstructured Problems. *Concurr. Comp.-Pract. E.*, 6(2):101–117, 1994.

[3] M. Blondi and M. Bondanza. Parallelizzazione di un Algoritmo per la Risoluzione del Problema del Commesso Viaggiatore. Master's thesis, Politecnico di Milano, 1993.

[4] R.D. Cook, D.S. Malkus, M.E. Plesha, and R.J. Witt. *Concepts and Applications of Finite Element Analysis*. John Wiley & Sons, 2001.

[5] M. Dorigo, G. Di Caro. The Ant Colony Optimization Meta-Heuristic. In D. Corne, M. Dorigo, and F. Glover, editors, *New Ideas in Optimization*, McGraw-Hill, 1999.

[6] M. Dorigo. Optimization, Learning and Natural Algorithms. PhD Thesis, Dipartimento di Elettronica, Politecnico di Milano, 1992.

[7] C.M. Fiduccia and R.M. Mattheyses. A Linear Time Heuristic for Improving Network Partitions. In *Proc. 19th IEEE Design Automation Conf.*, Las Vegas, NV, 1982, pages 175–181.

[8] Graph Collection. `wwwcs.uni-paderborn.de/cs/ag-monien/RESEARCH/PART/graphs.html`.

[9] B. Hendrickson and R. Leland. A Multilevel Algorithm for Partitioning Graphs. In *Proc. ACM/IEEE Conf. Supercomputing*, San Diego, CA, 1995.

[10] P. Kadłuczka and K. Wala. Tabu Search and Genetic Algorithms for the Generalized Graph Partitioning Problem. *Control Cybern.*, 24(4)459–476, 1995.

[11] G. Karypis and V. Kumar. Multilevel k-way Partitioning Scheme for Irregular Graphs. *J. Parallel Distr. Com.*, 48(1):96–129, 1998.

[12] B.W. Kernighan and S. Lin. An Efficient Heuristic Procedure for Partitioning Graph. *Bell Sys. Tech. J.*, 49(2)291–307, 1970.

[13] A.E. Langham and P.W. Grant. Using Competing Ant Colonies to Solve k-way Partitioning Problems with Foraging and raiding strategies. *Lect. Notes Comp. Sc.*, 1674:621–625, 1999.

[14] P. Korošec, J. Šilc, and B. Robič. Solving the Mesh-partitioning Problem with an Ant-colony Algorithm. *Parallel Comput.*, 30(5-6):785–801, 2004

[15] M. Randall and A. Lewis. A Parallel Implementation of Ant Colony Optimization. *J. Parallel Distr. Com.*, 62(9):1421–1432, 2002.

[16] T. Stützle. Parallelization Strategies for Ant Colony Optimization. *Lect. Notes Comp. Sc.*, 1498:722–741, 1998.

[17] C. Walshaw and M. Cross. Mesh Partitioning: A Multilevel Balancing and Refinement Algorithm. *SIAM J. Sci. Comput.*, 22(1):63–80, 2001.

Solving Engineering Optimization Problems with the Simple Constrained Particle Swarm Optimizer

Leticia C. Cagnina and Susana C. Esquivel
LIDIC, Universidad Nacional de San Luis, San Luis, Argentina
E-mail: lcagnina@unsl.edu.ar

Carlos A. Coello Coello
CINVESTAV-IPN, Mexico D. F., Mexico

Keywords: constrained optimization, engineering problems, particle swarm optimization

Received: July 7, 2008

This paper introduces a particle swarm optimization algorithm to solve constrained engineering optimization problems. The proposed approach uses a relatively simple method to handle constraints and a different mechanism to update the velocity and position of each particle. The algorithm is validated using four standard engineering design problems reported in the specialized literature and it is compared with respect to algorithms representative of the state-of-the-art in the area. Our results indicate that the proposed scheme is a promising alternative to solve this sort of problems because it obtains good results with a low number of objective functions evaluations.

Povzetek: Članek uvaja za reševanje inženirskih optimizacijskih problemov z omejitvami algoritem za optimizacijo z roji.

1 Introduction

Engineering design optimization problems are normally adopted in the specialized literature to show the effectiveness of new constrained optimization algorithms. These nonlinear engineering problems have been investigated by many researchers that used different methods to solve them: Branch and Bound using SQP [24], Recursive Quadratic Programming [9], Sequential Linearization Algorithm [20], Integer-discrete-continuous Nonlinear Programming [11], Nonlinear Mixed-discrete Programming [19], Simulated Annealing [27], Genetic Algorithms [26], Evolutionary Programming [8] and, Evolution Strategies [25] among many others. These types of problems normally have mixed (e.g., continuous and discrete) design variables, nonlinear objective functions and nonlinear constraints, some of which may be active at the global optimum. Constraints are very important in engineering design problems, since they are normally imposed on the statement of the problems and sometimes are very hard to satisfy, which makes the search difficult and inefficient.

Particle Swarm Optimization (PSO) is a relatively recent bio-inspired metaheuristic, which has been found to be highly competitive in a wide variety of optimization problems. However, its use in engineering optimization problems and in constrained optimization problems, in general, has not been as common as in other areas (e.g., for adjusting weights in a neural network). The approach described in this paper contains a constraint-handling technique as well as a mechanism to update the velocity and position of the particles, which is different from the one adopted by the original PSO.

This paper is organized as follows. Section 2 briefly discusses the previous related work. Section 3 describes in detail our proposed approach. Section 4 presents the experimental setup adopted and provides an analysis of the results obtained from our empirical study. Our conclusions and some possible paths for future research are provided in Section 5.

2 Literature review

Guo et al. presented a hybrid swarm intelligent algorithm with an improvement in global search reliability. They tested the algorithm with two of the problems adopted here (E02 and E04). Despite they claim that their algorithm is superior for finding the best solutions (in terms of quality and robustness), the solution that they found for E02 is greater than its best known value and for E04 the results obtained are not comparable to ours, because they used more constraints in the definition of that problem [13].

Shamim et al. proposed a method based on a socio-behavioral simulation model. The idea behind this approach is that the leaders of all societies interact among themselves for the improvement of the society. They tested their algorithm using three of the problems adopted here (E01, E02 and E03). The best values reported for these three problems are close from the optimal known values. The number of fitness function evaluations was 19,259 for

E01, 19,154 for E02 and 12,630 for E03 [1].

Mahdavi et al. developed an improved harmony search algorithm with a novel method for generating new solutions that enhances the accuracy and the convergence rate of the harmony search. They used three of the problems adopted here (E01, E03 and E04) to validate their approach, performing 300,000, 200,000 and 50,000 evaluations, respectively. For E01 and E02, the best values reported are not the best known values because the ranges of some variables in E01 are different from those of the original description of the problem (x_4 is out of range), which makes such solution infeasible under the description adopted here. The value reported by them for E04 is very close to the best value known [21].

Bernardino et al. hybridized a genetic algorithm embedding an artificial immune system into its search engine, in order to help moving the population into the feasible region. The algorithm was used to solve four of the test problems adopted here (E01, E02, E03 and E04), using 320,000, 80,000, 36,000 and 36,000 evaluations of the objective functions, respectively. The best values found for E01, E02 and E04 are close to the best known. For E03 the value reported is better than the best known, because one of the decision variables is out of range (x_5). The values in general, are good, although the number of evaluations required to obtain them is higher than those required by other algorithms [4].

Hernandez Aguirre et al. proposed a PSO algorithm with two new perturbation operators aimed to prevent premature convergence, as well as a new neighborhood structure. They used an external file to store some particles and, in that way, extend their life after the adjustment of the tolerance of the constraints. The authors reference three algorithms which obtained good results for the problems adopted in their study: two PSO-based algorithms and a Differential Evolution (DE) algorithm. One of the PSO-based approaches compared [16] used three of the problems adopted here (E01, E02 and E04), performing 200,000 objective function evaluations. The other PSO-based approach compared [14] was tested with the same set of problems and the best known values were reached for E02 and E04 after 30,000 objective function evaluations. The DE algorithm [22] reported good results with 30,000 evaluations for the four problems. This same number of evaluations was performed by the algorithm proposed by Hernandez et al. and their results are the best reported until now for the aforementioned problems [15].

For that reason, we used these last two algorithms to compare the performance of our proposed approach. The DE algorithm [22] will be referenced as “Mezura” and, the PSO by [15] as “COPSO”.

3 Our proposed approach: SiC-PSO

The particles in our proposed approach (called ***S**imple **C**onstrained **P**article **S**warm **O**ptimizer*, or SiC-PSO), are n-dimensional values (continuous, discrete or a combination of both) vectors, where n refers to the number of decision variables of the problem to be solved. Our approach adopts one of the most simple constraint-handling methods currently available. Particles are compared by pairs: 1) if the two particles are feasible, we choose the one with a better fitness function value; 2) if the two particles are infeasible, we choose the particle with the lower infeasibility degree; 3) if one particle is feasible and the other is infeasible, we choose the feasible one. This strategy is used when the *pbest*, *gbest* and *lbest* particles are chosen. When an individual is found infeasible, the amount of violation (this value is normalized with respect to the largest violation stored so far) is added. So, each particle saves its infeasibility degree reached until that moment.

As in the basic PSO [10], our proposed algorithm records the best position found so far for each particle (*pbest* value) and, the best position reached by any particle into the swarm (*gbest* value). In other words, we adopt the *gbest* model. But in previous works, we found that the *gbest* model tends to converge to a local optimum very often [7]. Motivated by this, we proposed a formula to update the velocity, using a combination of both the *gbest* and the *lbest* models [5]. Such a formula (Eq. 1) is adopted here as well. The *lbest* model is implemented using a ring topology [17] to calculate the neighborhoods of each particle. For a size of neighborhood of three particles and a swarm of six particles (1,2,3,4,5,6), the neighborhoods considered are the following: (1,2,3), (2,3,4), (3,4,5), (4,5,6), (5,6,1) and (6,1,2). The formula for updating particles is the same that in the basic PSO and it is shown in Eq. 2.

$$\begin{aligned} v_{id} &= w(v_{id} + c_1 r_1 (pb_{id} - p_{id}) \\ &\quad + c_2 r_2 (pl_{id} - p_{id}) \\ &\quad + c_3 r_3 (pg_d - p_{id})) \end{aligned} \tag{1}$$

$$p_{id} = p_{id} + v_{id} \tag{2}$$

where v_{id} is the velocity of the particle i at the dimension d, w is the inertia factor [10] whose goal is to balance the global exploration and the local exploitation, c_1 is the personal learning factor, and c_2, c_3 are the social learning factors, r_1, r_2 and r_3 are three random numbers within the range [0..1], pb_{id} is the best position reached by the particle i, pl_{id} is the best position reached by any particle in the neighborhood of particle i and, pg_d is the best position reached by any particle in the swarm. Finally, p_{id} is the value of the particle i at the dimension d.

We empirically found that for some difficult functions, a previous version of our algorithm could not find good values. The reason was its diversification of solutions which kept the approach from converging. In SiC-PSO we changed the common updating formula (Eq. 2) of the particles for the update equation presented by Kennedy [18]. In Kennedy’s algorithm, the new position of each particle is randomly chosen from a Gaussian distribution with the mean selected as the average between the best position recorded for the particle and the best in its neighborhood.

The standard deviation is the difference between these two values. We adapted that formula adding the global best (*gbest*) to the best position of the particle and the best in its neighborhood. We also changed the way in which the standard deviation is determined. We use the *pbest* and, the *gbest* instead of the *lbest* as was proposed by Kennedy. We determined those changes after several empirical tests with different Gaussian random generator parameters. Thus, the position is updated using the following equation:

$$p_i = N\left(\frac{p_i + pl_i + pg}{3}, |p_i - pg|\right) \qquad (3)$$

where p_i, pl_i and pg are defined as before and, N is the value returned by the Gaussian random generator. SiC-PSO used Eq. 3 and Eq. 2 for the updating of positions of the particles. We considered a high probability for selecting Eq. 3 (0.925) over Eq. 2. We chose that probability after conducting numerous experiments.

4 Parameter settings and analysis of results

A set of 4 engineering design optimization problems was chosen to evaluate the performance of our proposed algorithm. A detailed description of the test problems may be consulted in the appendix at the end of this paper. We performed 30 independent runs per problem, with a total of 24,000 objective function evaluations per run. We also tested the algorithm with 27,000 and 30,000 evaluations of the objective function, but no performance improvements were noticed in such cases. Our algorithm used the following parameters: swarm size = 8 particles, neighborhood size = 3, inertia factor w = 0.8, personal learning factor and social learning factors for c_1, c_2 and c_3 were set to 1.8. These parameter settings were empirically derived after numerous previous experiments.

Our results were compared with respect to the best results reported in the specialized literature. Those values were obtained by Hernandez Aguirre et al. [15] and Mezura et al. [22]. We reference those results into the tables shown next as “COPSO” and “Mezura”, respectively. It is important remark that COPSO and Mezura algorithms reached the best values after 30,000 fitness function evaluations, which is a larger value than that required by our algorithm. The best values are shown in Table 1 and, the mean and standard deviations over the 30 runs are shown in Table 2.

The three algorithms reached the best known values for E01. For E02, SiC-PSO and COPSO reached the best known, but Mezura reported a value with a precision of only 4 digits after the decimal point, and the exact value reached by them is not reported. For E03, SiC-PSO reached the best value, COPSO reached a value slightly worse than ours, and Mezura reached an infeasible value. SiC-PSO and COPSO reached the best value for E04, although Mezura reported a value that is worse than the best known. In general, COPSO obtained the best mean values, except for E03 for which best mean was found by our algorithm. The lower standard deviation values for E01 and E04 was obtained by COPSO; for E02 and E03, our SiC-PSO found the minimum values.

Tables 3, 4, 5 and 6 show the solution vectors of the best solution reached by SiC-PSO as well as the values of the constraints, for each of the problems tested.

	Best Solution
x_1	0.205729
x_2	3.470488
x_3	9.036624
x_4	0.205729
$g_1(\vec{x})$	-1.819E-12
$g_2(\vec{x})$	-0.003721
$g_3(\vec{x})$	0.000000
$g_4(\vec{x})$	-3.432983
$g_5(\vec{x})$	-0.080729
$g_6(\vec{x})$	-0.235540
$g_7(\vec{x})$	0.000000
$f(\vec{x})$	1.724852

Table 3: SiC-PSO Solution vector for E01 (welded beam).

	Best Solution
x_1	0.812500
x_2	0.437500
x_3	42.098445
x_4	176.636595
$g_1(\vec{x})$	-4.500E-15
$g_2(\vec{x})$	-0.035880
$g_3(\vec{x})$	-1.164E-10
$g_4(\vec{x})$	-63.363404
$f(\vec{x})$	6,059.714335

Table 4: SiC-PSO Solution vector for E02 (pressure vessel).

5 Conclusions and Future Work

We have presented a simple PSO algorithm (SiC-PSO) for constrained optimization problems. The proposed approach uses a simple constraint-handling mechanism, a ring topology for implementing the *lbest* model and a novel formula to update the position of particles. SiC-PSO had a very good performance when is applied to several engineering design optimization problems. We compared our results with respect to those obtained by two algorithms that had been previously found to perform well in the same problems. These two algorithms are more sophisticated than our SiC-PSO. Our algorithm obtained the optimal values for each of the test problems studied, while performing a lower number of objective function evaluations. Also, the performance of our approach with respect to the mean and standard deviation is comparable with that shown by the

Prob.	Optimal	SiC−PSO	COPSO	Mezura
E01	1.724852	1.724852	1.724852	1.724852
E02	6,059.714335	6,059.714335	6,059.714335	6,059.7143
E03	NA	2,996.348165	2,996.372448	2,996.348094*
E04	0.012665	0.012665	0.012665	0.012689

*Infeasible solution. NA Not avaliable.

Table 1: Best results obtained by SiC-PSO, COPSO and Mezura.

	Mean			St. Dev.		
Prob.	SiC−PSO	COPSO	Mezura	SiC−PSO	COPSO	Mezura
E01	2.0574	1.7248	1.7776	0.2154	1.2E-05	8.8E-02
E02	6,092.0498	6,071.0133	6,379.9380	12.1725	15.1011	210.0000
E03	2,996.3482	2,996.4085	2,996.3480*	0.0000	0.0286	0.0000*
E04	0.0131	0.0126	0.0131	4.1E-04	1.2E-06	3.9E-04

*Infeasible solution.

Table 2: Means and Standard Deviations for the results obtained.

	Best Solution
x_1	3.500000
x_2	0.700000
x_3	17
$x4$	7.300000
x_5	7.800000
x_6	3.350214
x_7	5.286683
$g_1(\vec{x})$	-0.073915
$g_2(\vec{x})$	-0.197998
$g_3(\vec{x})$	-0.499172
$g_4(\vec{x})$	-0.901471
$g_5(\vec{x})$	0.000000
$g_6(\vec{x})$	-5.000E-16
$g_7(\vec{x})$	-0.702500
$g_8(\vec{x})$	-1.000E-16
$g_9(\vec{x})$	-0.583333
$g_{10}(\vec{x})$	-0.051325
$g_{11}(\vec{x})$	-0.010852
$f(\vec{x})$	2,996.348165

Table 5: SiC-PSO Solution vector for E03 (speed reducer).

other algorithms. Thus, we consider our approach to be a viable choice for solving constrained engineering optimization problems, due to its simplicity, speed and reliability. As part of our future work, we are interested in exploring other PSO models and in performing a more detailed statistical analysis of the performance of our proposed approach.

Appendix: Engineering problems

Formulating of the engineering design problems used to test the algorithm proposed.

	Best Solution
x_1	0.051583
x_2	0.354190
x_3	11.438675
$g_1(\vec{x})$	-2.000E-16
$g_2(\vec{x})$	-1.000E-16
$g_3(\vec{x})$	-4.048765
$g_4(\vec{x})$	-0.729483
$f(\vec{x})$	0.012665

Table 6: SiC-PSO Solution vector for E04 (tension/compression spring).

E01: Welded beam design optimization problem

The problem is to design a welded beam for minimum cost, subject to some constraints [23]. Figure 1 shows the welded beam structure which consists of a beam A and the weld required to hold it to member B. The objective is to find the minimum fabrication cost, considerating four design variables: x_1, x_2, x_3, x_4 and constraints of shear stress τ, bending stress in the beam σ, buckling load on the bar P_c, and end deflection on the beam δ. The optimization model is summarized in the next equation:

Minimize:

$$f(\vec{x}) = 1.10471{x_1}^2x_2 + 0.04811x_3x_4(14.0 + x_2)$$

subject to:

$$g_1(\vec{x}) = \tau(\vec{x}) - 13,600 \leq 0$$

$$g_2(\vec{x}) = \sigma(\vec{x}) - 30,000 \leq 0$$

$$g_3(\vec{x}) = x_1 - x_4 \leq 0$$

$$g_4(\vec{x}) = 0.10471({x_1}^2) + 0.04811x_3x_4(14 + x_2) - 5.0 \leq 0$$

$$g_5(\vec{x}) = 0.125 - x_1 \leq 0$$

$$g_6(\vec{x}) = \delta(\vec{x}) - 0.25 \leq 0$$

$$g_7(\vec{x}) = 6,000 - Pc(\vec{x}) \leq 0$$

with:

$$\tau(\vec{x}) = \sqrt{(\tau')^2 + (2\tau'\tau'')\frac{x_2}{2R} + (\tau'')^2}$$

$$\tau' = \frac{6,000}{\sqrt{2}x_1x_2}$$

$$\tau'' = \frac{MR}{J}$$

$$M = 6,000\left(14 + \frac{x_2}{2}\right)$$

$$R = \sqrt{\frac{{x_2}^2}{4} + \left(\frac{x_1 + x_3}{2}\right)^2}$$

$$J = 2\left\{x_1x_2\sqrt{2}\left[\frac{{x_2}^2}{12} + \left(\frac{x_1 + x_3}{2}\right)^2\right]\right\}$$

$$\sigma(\vec{x}) = \frac{504,000}{x_4{x_3}^2}$$

$$\delta(\vec{x}) = \frac{65,856,000}{(30 \times 10^6)x_4{x_3}^3}$$

$$Pc(\vec{x}) = \frac{4.013(30 \times 10^6)\sqrt{\frac{{x_3}^2{x_4}^6}{36}}}{196}\left(1 - \frac{x_3\sqrt{\frac{30\times10^6}{4(12\times10^6)}}}{28}\right)$$

with $0.1 \leq x_1, x_4 \leq 2.0$, and $0.1 \leq x_2, x_3 \leq 10.0$.
Best solution:

$$x^* = (0.205730, 3.470489, 9.036624, 0.205729)$$

where $f(x^*) = 1.724852$.

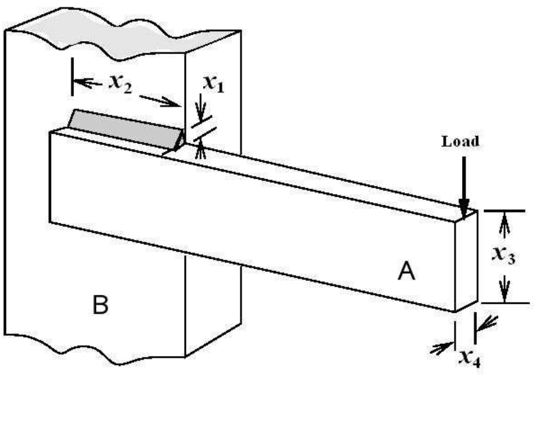


Figure 1: Weldem Beam.

E02: Pressure Vessel design optimization problem

A compressed air storage tank with a working pressure of 3,000 psi and a minimum volume of 750 ft^3. A cylindrical vessel is capped at both ends by hemispherical heads (see Fig. 2). Using rolled steel plate, the shell is made in two halves that are joined by teo longitudinal welds to form a cylinder. The objective is minimize the total cost, including the cost of the materials forming the welding [24]. The design variables are: thickness x_1, thickness of the head x_2, the inner radius x_3, and the length of the cylindrical section of the vessel x_4. The variables x_1 and x_2 are discrete values which are integer multiples of 0.0625 inch. Then, the formal statement is:
Minimize:

$$\begin{aligned} f(\vec{x}) &= 0.6224x_1x_3x_4 + 1.7781x_2{x_3}^2 + 3.1661{x_1}^2x_4 \\ &+ 19.84{x_1}^2x_3 \end{aligned}$$

subject to:

$$g_1(\vec{x}) = -x_1 + 0.0193x_3 \leq 0$$

$$g_2(\vec{x}) = -x_2 + 0.00954x_3 \leq 0$$

$$g_3(\vec{x}) = -\pi{x_3}^2{x_4}^2 - \frac{4}{3}\pi{x_3}^3 + 1,296,000 \leq 0$$

$$g_4(\vec{x}) = x_4 - 240 \leq 0$$

with $1 \times 0.0625 \leq x_1, x_2 \leq 99 \times 0.0625, 10.0 \leq x_3$, and $x_4 \leq 200.0$.

Best solution:

$$x^* = (0.8125, 0.4375, 42.098446, 176.636596)$$

where $f(x^*) = 6,059.714335$.

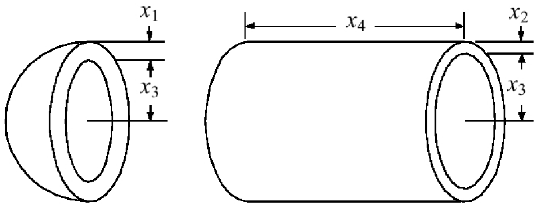


Figure 2: Pressure Vessel.

E03: Speed Reducer design optimization problem

The design of the speed reducer [12] shown in Fig. 3, is considered with the face width x_1, module of teeth x_2, number of teeth on pinion x_3, length of the first shaft between bearings x_4, length of the second shaft between bearings x_5, diameter of the first shaft x_6, and diameter of the first shaft x_7 (all variables continuous except x_3 that is integer). The weight of the speed reducer is to be minimized subject to constraints on bending stress of the gear teeth, surface stress, transverse deflections of the shafts and stresses in the shaft. The problem is:

Minimize:

$$\begin{aligned} f(\vec{x}) &= 0.7854x_1{x_2}^2(3.3333x_3^2 + 14.9334x_3 - 43.0934) \\ &- 1.508x_1(x_6^2 + x_7^2) + 7.4777(x_6^3 + x_7^3) \\ &+ 0.7854(x_4x_6^2 + x_5x_7^2) \end{aligned}$$

subject to:

$$g_1(\vec{x}) = \frac{27}{x_1x_2^2x_3} - 1 \leq 0$$

$$g_2(\vec{x}) = \frac{397.5}{x_1 x_2^2 x_3^2} - 1 \le 0$$

$$g_3(\vec{x}) = \frac{1.93 x_4^3}{x_2 x_3 x_6^4} - 1 \le 0$$

$$g_4(\vec{x}) = \frac{1.93 x_5^3}{x_2 x_3 x_7^4} - 1 \le 0$$

$$g_5(\vec{x}) = \frac{1.0}{110 x_6^3}\sqrt{\left(\frac{745.0 x_4}{x_2 x_3}\right)^2 + 16.9 \times 10^6} - 1 \le 0$$

$$g_6(\vec{x}) = \frac{1.0}{85 x_7^3}\sqrt{\left(\frac{745.0 x_5}{x_2 x_3}\right)^2 + 157.5 \times 10^6} - 1 \le 0$$

$$g_7(\vec{x}) = \frac{x_2 x_3}{40} - 1 \le 0$$

$$g_8(\vec{x}) = \frac{5 x_2}{x_1} - 1 \le 0$$

$$g_9(\vec{x}) = \frac{x_1}{12 x_2} - 1 \le 0$$

$$g_{10}(\vec{x}) = \frac{1.5 x_6 + 1.9}{x_4} - 1 \le 0$$

$$g_{11}(\vec{x}) = \frac{1.1 x_7 + 1.9}{x_5} - 1 \le 0$$

with $2.6 \le x_1 \le 3.6, 0.7 \le x_2 \le 0.8, 17 \le x_3 \le 28, 7.3 \le x_4 \le 8.3, 7.8 \le x_5 \le 8.3, 2.9 \le x_6 \le 3.9$, and $5.0 \le x_7 \le 5.5$.

Best solution:

$$x^* = (3.500000, 0.7, 17, 7.300000, 7.800000, 3.350214, 5.286683)$$

where $f(x^*) = 2,996.348165$.

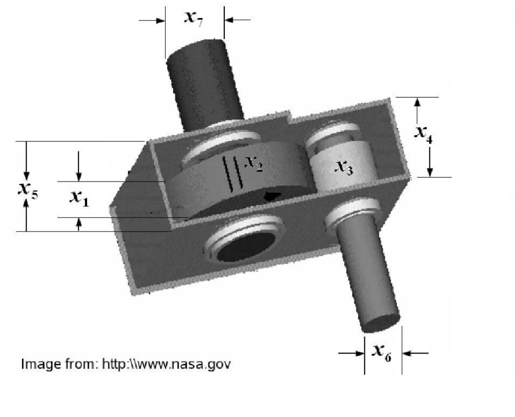


Figure 3: Speed Reducer.

5.1 E04: Tension/compression spring design optimization problem

This problem [2] [3] minimizes the weight of a tension/compression spring (Fig. 4), subject to constraints of minimum deflection, shear stress, surge frequency, and limits on outside diameter and on design variables. There are three design variables: the wire diameter x_1, the mean coil diameter x_2, and the number of active coils x_3. The mathematical formulation of this problem is:

Minimize:

$$f(\vec{x}) = (x_3 + 2) x_2 x_1^2$$

subject to:

$$g_1(\vec{x}) = 1 - \frac{x_2^3 x_3}{7,178 x_1^4} \le 0$$

$$g_2(\vec{x}) = \frac{4x_2^2 - x_1 x_2}{12,566(x_2 x_1^3) - x_1^4} + \frac{1}{5,108 x_1^2} - 1 \le 0$$

$$g_3(\vec{x}) = 1 - \frac{140.45 x_1}{x_2^2 x_3} \le 0$$

$$g_4(\vec{x}) = \frac{x_2 + x_1}{1.5} - 1 \le 0$$

with $0.05 \le x_1 \le 2.0, 0.25 \le x_2 \le 1.3$, and $2.0 \le x_3 \le 15.0$.

Best solution:

$$x^* = (0.051690, 0.356750, 11.287126)$$

where $f(x^*) = 0.012665$.

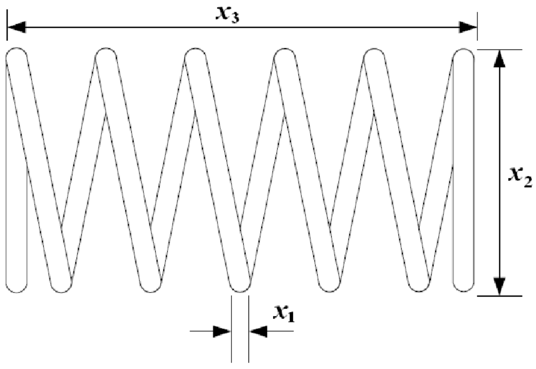


Figure 4: Tension/Compression Spring.

Acknowledgment

The first and second author acknowledge support from the ANPCyT (National Agency to Promote Science and Technology, PICT 2005 and Universidad Nacional de San Luis. The third author acknowledges support from CONACyT project no. 45683-Y.

References

[1] S. Akhtar, K. Tai and T. Ray. A Socio-behavioural Simulation Model for Engineering Design Optimization. *Eng. Optimiz.*, 34(4):341–354, 2002.

[2] J. Arora. *Introduction to Optimum Design*. McGraw-Hill, 1989.

[3] A. Belegundu. *A Study of Mathematical Programming Methods for Structural Optimization*. PhD thesis, Department of Civil Environmental Engineering, University of Iowa, Iowa, 1982.

[4] H. Bernardino, H. Barbosa and A. Lemonge. A Hybrid Genetic Algorithm for Constrained Optimization Problems in Mechanical Engineering. In *Proc. IEEE Congress on Evolutionary Computation (CEC 2007)*, Singapore, 2007, pages 646–653.

[5] L. Cagnina, S. Esquivel and C. Coello Coello. A Particle Swarm Optimizer for Constrained Numerical Optimization. In *Proc. 9th International Conference on Parallel Problem Solving from Nature (PPSN IX)*, Reykjavik, Iceland, 2006, pages 910–919.

[6] L. Cagnina, S. Esquivel and C. Coello Coello. A Bi-population PSO with a Shake-Mechanism for Solving Constrained Numerical Optimization. In *Proc. IEEE Congress on Evolutionary Computation (CEC2007)*, Singapore, 2007, pages 670–676.

[7] L. Cagnina, S. Esquivel and R. Gallard. Particle Swarm Optimization for Sequencing Problems: a Case Study. In *Proc. IEEE Congress on Evolutionary Computation (CEC 2004)*, Portland, Oregon, USA, 2004, pages 536–541.

[8] Y. Cao and Q. Wu. Mechanical Design Optimization by Mixed-variable Evolutionary Programming. In *1997 IEEE International Conference on Evolutionary Computation*, Indianapolis, Indiana, USA, 1997, pages 443–446.

[9] J. Cha and R. Mayne. Optimization with Discrete Variables via Recursive Quadratic Programming: part II. *J. Mech. Transm.-T. ASME*, 111(1):130–136, 1989.

[10] R. Eberhart and Y. Shi. A Modified Particle Swarm Optimizer. In *Proc. IEEE International Conference on Evolutionary Computation*, Anchorage, Alaska, USA, 1998, pages 69–73.

[11] J. Fu, R. Fenton and W. Cleghorn. A Mixed Integer-discrete-continuous Programming Method and its Applications to Engineering Design Optimization. *Eng. Optimiz.*, 17(4):263–280, 1991.

[12] J. Golinski. An Adaptive Optimization System Applied to Machine Synthesis. *Mech. Mach. Theory*, 8(4):419Ű436, 1973.

[13] C. Guo, J. Hu, B. Ye and Y. Cao. Swarm Intelligence for Mixed-variable Design Optimization. *J. Zheijiang University Science*, 5(7):851–860, 1994.

[14] S. He, E. Prempain and Q. Wu. An Improved Particle Swarm Optimizer for Mechanical Design Optimization Problems. *Eng. Optimiz.*, 36(5):585–605, 2004.

[15] A. Hernandez Aguirre, A. Muñoz Zavala, E. Villa Diharce and S. Botello Rionda. COPSO: Constrained Optimization via PSO Algorithm. Technical report No. I-07-04/22-02-2007, Center for Research in Mathematics (CIMAT), 2007.

[16] X. Hu, R. Eberhart and Y. Shi. Engineering Optimization with Particle Swarm. In *Proc. IEEE Swarm Intelligence Symposium*, Indianapolis, Indiana, USA, 2003, pages 53–57.

[17] J. Kennedy. Small World and Mega-Minds: Effects of Neighborhood Topologies on Particle Swarm Performance. In *IEEE Congress on Evolutionary Computation (CEC 1999)*, Washington, DC, USA, 1999, pages 1931–1938.

[18] J. Kennedy and R. Eberhart. Bores Bones Particle Swarm. In *Proc. IEEE Swarm Intelligence Symposium*, Indianapolis, Indiana, USA, 2003, pages 80–89.

[19] H. Li and T. Chou. A Global Approach of Nonlinear Mixed Discrete Programming in Design Optimization. *Eng. Optimiz.*, 22(2):109–122, 1993.

[20] H. Loh and P. Papalambros. A Sequential Linearization Approach for Solving Mixed-discrete Nonlinear Design Optimization Problems. *J. Mech. Des.-T. ASME*, 113(3):325–334, 1991.

[21] M. Mahdavi, M. Fesanghary and E. Damangir. An Improved Harmony Search Algorithm for Solving Optimization Problems. *Appl. Math. Comput.*, 188(2):1567–1579, 2007.

[22] E. Mezura and C. Coello. Useful Infeasible Solutions in Engineering Optimization with Evolutionary Algorithms. *Lect. Notes Comput. Sc.*, 3789:652–662, 2005.

[23] K. Ragsdell and D. Phillips. Optimal Design of a Class of Welded Structures using Geometric Programming. *J. Eng. Ind.*, 98(3):1021–1025, 1976.

[24] E. Sandgren. Nonlinear Integer and Discrete Programming in Mechanical Design Optimization. *J. Mech. Des.-T. ASME*, 112(2):223–229, 1990.

[25] G. Thierauf and J. Cai. Evolution Strategies - Parallelization and Applications in Engineering Optimization. In *Parallel and Distributed Precessing for Computational Mechanics.* B.H.V. Topping (ed.), Saxe-Coburg Publications, 2000, pages 329–349.

[26] S. Wu and T. Chou. Genetic Algorithms for Nonlinear Mixed Discrete-integer Optimization Problems via Meta-genetic Parameter Optimization. *Eng. Optimiz.*, 24(2):137–159, 1995.

[27] C. Zhang and H. Wang. Mixed-discrete Nonlinear Optimization with Simulated Annealing. *Eng. Optimiz.*, 21(4):277–291, 1993.

Balancing Load in a Computational Grid Applying Adaptive, Intelligent Colonies of Ants

Mohsen Amini Salehi
Department of Software Engineering, Faculty of Engineering,
Islamic Azad University, Mashhad Branch, Iran
E-mail: Amini@mshdiau.ac.ir

Hossein Deldari
Department of Software Engineering, Faculty of Engineering
Ferdowsi University of Mashhad, Iran
E-mail: hd@um.ac.ir

Bahare Mokarram Dorri
Management and Planning Organisation of Khorasan, Mashhad, Iran
E-mail: mokarram@mpo-kh.ir

Keywords: grid computing, load balancing, ant colony, agent-based resource management system (ARMS)

Received: July 1, 2007

Load balancing is substantial when developing parallel and distributed computing applications. The emergence of computational grids extends the necessity of this problem. Ant colony is a meta-heuristic method that can be instrumental for grid load balancing. This paper presents an echo system of adaptive fuzzy ants. The ants in this environment can create new ones and may also commit suicide depending on existing conditions. A new concept called Ant level load balancing is presented here for improving the performance of the mechanism. A performance evaluation model is also derived. Then theoretical analyses, which are supported with experiment results, prove that this new mechanism surpasses its predecessor.

Povzetek: Za porazdeljevanje obremenitev je predlagana nova metoda s kolonijami mravelj.

1 Introduction

A computational grid is a hardware and software infrastructure which provides consistent, pervasive and inexpensive access to high end computational capacity. An ideal grid environment should provide access to all the available resources seamlessly and fairly.

The resource manager is an important infrastructural component of a grid computing environment. Its overall aim is to efficiently schedule applications needing utilization of available resources in the grid environment. A grid resource manager provides a mechanism for grid applications to discover and utilize resources in the grid environment. Resource discovery and advertisement offer complementary functions. The discovery is initiated by a grid application to find suitable resources within the grid. Advertisement is initiated by a resource in search of a suitable application that can utilize it. A matchmaker is a grid middleware component which tries to match applications and resources. A matchmaker may be implemented in centralized or distributed ways. As the grid is inherently dynamic, and has no boundary [1], so the distributed approaches usually show better results [2] and are also more scalable. A good matchmaker (broker) should uniformly distribute the requests, along the grid resources, with the aid of load balancing methods.

As mentioned in [1], the grid is a highly dynamic environment for which there is no unique administration. Therefore, the grid middleware should compensate for the lack of unique administration.

ARMS is an agent-based resource manager infrastructure for the grid [3, 4]. In ARMS, each agent can act simultaneously as a resource questioner, resource provider, and the matchmaker. Details of the design and implementation of ARMS can be found in [2]. In this work, we use ARMS as the experimental platform.

Cosy is a job scheduler which supports job scheduling as well as advanced reservations [5]. It is integrated into ARMS agents to perform global grid management [5]; Cosy needs a load balancer to better utilize available resources. This load balancer is introduced in part 3.

The rest of the paper is organized as follows: Section 2 introduces the load balancing approaches for grid resource management. In Section 3, ant colony optimization and self-organizing mechanisms for load balancing are discussed. Section 4 describes the proposed mechanism. Performance metrics and simulation results are included in Section 5. Finally, the conclusion of the article is presented as well as future work related to this research.

2 Load balancing

Load balancing algorithms are essentially designed to spread the resources' load equally thus maximizing their utilization while minimizing the total task execution time [7]. This is crucial in a computational grid where the most pressing issue is to fairly assign jobs to resources. Thus, the difference between the heaviest and the lightest resource load is minimized.

A flexible load sharing algorithm is required to be general, adaptable, stable, scalable, fault tolerant, transparent to the application and to also induce minimum overhead to the system [8]. The properties listed above are interdependent. For example, a lengthy delay in processing and communication can affect the algorithm overhead significantly, result in instability and indicate that the algorithm is not scalable.

The load balancing process can be defined in three rules: the location, distribution and selection rule [7]. The location rule determines which resource domain will be included in the balancing operation. The domain may be local, i.e. inside the node, or global, i.e. between different nodes. The distribution rule establishes the redistribution of the workload among available resources in the domain, while the selection rule decides whether the load balancing operation can be performed preemptively or not [7].

2.1 Classification of load balancing mechanisms

In general, load balancing mechanisms can be broadly categorized as centralized or decentralized, dynamic or static [10], and periodic or non-periodic [11].

In a centralized algorithm, there is a central scheduler which gathers all load information from the nodes and makes appropriate decisions. However, this approach is not scalable for a vast environment like the grid. In decentralized models, there is usually not a specific node known as a server or collector. Instead, all nodes have information about some or all other nodes. This leads to a huge overhead in communication. Furthermore, this information is not very reliable because of the drastic load variation in the grid and the need to update frequently.

Static algorithms are not affected by the system state, as their behaviour is predetermined. On the other hand, dynamic algorithms make decisions according to the system state. The state refers to certain types of information, such as the number of jobs waiting in the ready queue, the current job arrival rate, etc [12]. Dynamic algorithms tend to have better performance than static ones [13].

Some dynamic load balancing algorithms are adaptive; in other words, dynamic policies are modifiable as the system state changes. Via this approach, methods adjust their activities based on system feedback [13].

3 Related works

Swarm intelligence [14] is inspired by the behaviour of insects, such as wasps, ants or honey bees. The ants, for example, have little intelligence for their hostile and dynamic environment [15]. However, they perform incredible activities such as organizing their dead in cemeteries and foraging for food. Actually, there is an indirect communication among ants which is achieved through their chemical substance deposits [16].

This ability of ants is applied in solving some heuristic problems, like optimal routing in a telecommunication network [15], coordinating robots, sorting [17], and especially load balancing [6, 9, 18, 19].

Messor [20] is the main contribution to the load balancing context.

3.1 Messor

Messor is a grid computing system that is implemented on top of the Anthill framework [18].

Ants in this system can be in Search–Max or Search–Min states. In the Search–Max state, an ant wanders around randomly until it finds an overloaded node. The ant then switches to the Search–Min state to find an underloaded node. After these states, the ant balances the two overloaded and underloaded nodes that it found. Once an ant encounters a node, it retains information about the nodes visited. Other ants which visit this node can apply this information to perform more efficiently. However, with respect to the dynamism of the grid, this information cannot be reliable for a long time and may even cause erroneous decision-making by other ants.

3.2 Self-Organizing agents for grid load balancing

In [6], J.Cao et al propose a self-organizing load balancing mechanism using ants in ARMS. As this mechanism is simple and inefficient, we call it the "seminal approach". The main purpose of this study is the optimization of this seminal mechanism. Thus, we propose a modified mechanism based on a swarm of intelligent ants that uniformly balance the load throughout the grid.

In this mechanism an ant always wanders '2m+ 1' steps to finally balance two overloaded and underloaded nodes.

As stated in [6], the efficiency of the mechanism highly depends on the number of cooperating ants (n) as well as their step count (m). If a loop includes a few steps, then the ant will initiate the load balancing process frequently, while if the ant starts with a larger m, then the frequency of performing load balancing decreases. This implies that the ant's step count should be determined according to the system load. However, with this method, the number of ants and the number of their steps are defined by the user and do not change during the load balancing process. In fact, defining the number of ants and their wandering steps by the user is impractical in an environment such as the grid, where users have no background knowledge and the ultimate goal is to introduce a transparent, powerful computing service to end users.

Considering the above faults, we propose a new mechanism that can be adaptive to environmental conditions and turn out better results. In the next section, the proposed method is described.

4 Proposed method

In the new mechanism, we propose an echo system of intelligent ants which react proportionally to their conditions. Interactions between these intelligent, autonomous ants result in load balancing throughout the grid.

In this case, an echo system creates ants on demand to achieve load balancing during their adaptive lives. They may bear offspring when they sense that the system is drastically unbalanced and commit suicide when they detect equilibrium in the environment. These ants care for every node visited during their steps and record node specifications for future decision making. Moreover, every ant in the new mechanism hops ‘m’ steps (the value of ‘m’ is determined adaptively) instead of ‘2m+1’. At the end of the ‘m’ steps, ‘k’ overloaded are equalized with ‘k’ underloaded nodes, in contrast to one overloaded with one underloaded according to the previous method. This results in an earlier convergence with fewer ants and less communication overhead.

In the next sections, the proposed method is described in more detail.

4.1 Creating ants

If a node understands that it is overloaded, it can create a new ant taking only a few steps to balance the load as quickly as possible. Actually, as referred in [2], neighbouring agents, in ARMS, exchange their load status periodically. If a node’s load is more than the average of its neighbours, for several periods of time, and it has not been visited by any ant during this time, then the node creates a new ant itself to balance its load throughout a wider area.

Load can be estimated several ways by an agent to distinguish whether a node is overloaded or not. For the sake of comparison with similar methods, the number of waiting jobs in a node is considered the criterion for load measurement.

4.2 Decision-making

Every ant is allocated to a memory space which records specifications of the environment while it wanders. The memory space is divided into an underloaded list (Min List) and an overloaded list (Max List). In the former, the ant saves specifications of the underloaded nodes visited. In the latter, specifications of the overloaded nodes visited are saved.

At every step, the ant randomly selects one of the node’s neighbours.

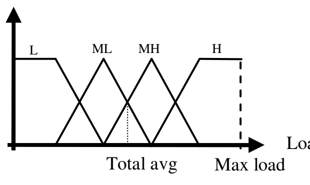
4.2.1 Deciding algorithm

After entering a node, the ant first checks its memory to determine whether this node was already visited by the ant itself or not. If not, the ant can verify the condition of the node, i.e. overloaded, underloaded or at an equilibrium, using its acquired knowledge from the environment.

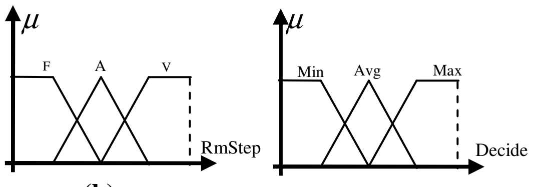
As the load quantity of a node is a linguistic variable and the state of the node is determined relative to system conditions, decision making is performed adaptively by applying fuzzy logic [21, 22].

To make a decision, the ant deploys the node’s current workload and the remaining steps as two inputs into the fuzzy inference system. Then, the ant determines the state of the node, i.e. Max, Avg or Min.

The total average of the load visited is kept as the ant’s internal knowledge about the environment. The ant uses this for building membership functions of the node’s workload, as shown in Figure1.a. The membership functions of Remain steps and Decide, as the output, are on the basis of a threshold and are presented in Figures 1.b, 1.c:



(a)



(b) (c)

Figure 1: Membership functions of fuzzy sets
a) The Node’s Load, b) Remain steps, c) Decide.

The inference system can be expressed as the following relation:

$$R_A : Load < L, ML, MH, H > * RmStep < F, A, V > \rightarrow Decide < Min, Avg, Max > \quad (1)$$

Where L, ML, MH, H in Figure1.a indicates Low, Medium Low, Medium High, High respectively and F, A, V in Figure 1.b indicates Few, Average and Very.

Thus, the ant can make a proper decision. If the result is “Max” or “Min”, the node’s specifications must be added to the ant’s max-list or the min-list. Subsequently, the corresponding counter for Max, Min, or Avg increases by one. These counters also depict the ant’s knowledge about the environment. How this knowledge is employed is explained in the next sections.

4.2.2 Ant level load balancing

In the subtle behaviour of ants and their interactions, we can see that when two ants face each other, they stop for a moment and touch tentacles, probably for recognizing their team members. This is what inspired the first use of ant level load balancing.

With respect to the system structure, it is probable that two or more ants meet each other on the same node. As mentioned earlier, each of these ants may gather specifications of some overloaded and underloaded nodes. The amount of information is not necessarily the same for each ant, for example one ant has specifications of four overloaded and two underloaded while the other has two overloaded and six underloaded nodes in the same position. In this situation, ants extend their knowledge by exchanging them. We call this *"ant level load balancing."* In the aforementioned example, after ant level load balancing of the two co-positions, the ants have specifications of three overloaded and four underloaded nodes in their memories. This leads to better performance in the last step, when the ants want to balance the load of 'k' overloaded with 'k' underloaded nodes. This operation can be applied to more than two ants.

Actually, when two or more co-positioned ants exchange their knowledge, they extend their movement radius to a bigger domain, thus improving awareness of the environment. Another idea is taken from the ant's pheromone deposits while travelling, which is used by ants to pursue other ants. This is applied in most ant colony optimization problems [23, 24]. There is, however, a subtle difference between these two applications. In the former the information retained by the ant may become invalid over time. This problem can be solved by evaporation [23, 24]. Evaporation, however, is not applicable in some cases, e.g. in the grid, where load information varies frequently. On the other hand, in the latter application, the knowledge exchanged is completely reliable.

4.2.3 How new ants are created

In special conditions, particularly when the its life span is long, the ant's memory may fill up, even though the ant may still be encountering nodes which are overloaded or underloaded. In this situation, if a node is overloaded, the ant bears a new ant with predefined steps. If encountering an underloaded node, the ant immediately exchanges the node's specification with the biggest load on the list of underloaded elements. This results in better balancing performance and adaptability to the environment. Here, adaptability translates into increasing the number of ants automatically, whenever there is an abundance of overloaded nodes.

4.3 Load balancing, starting new itineration

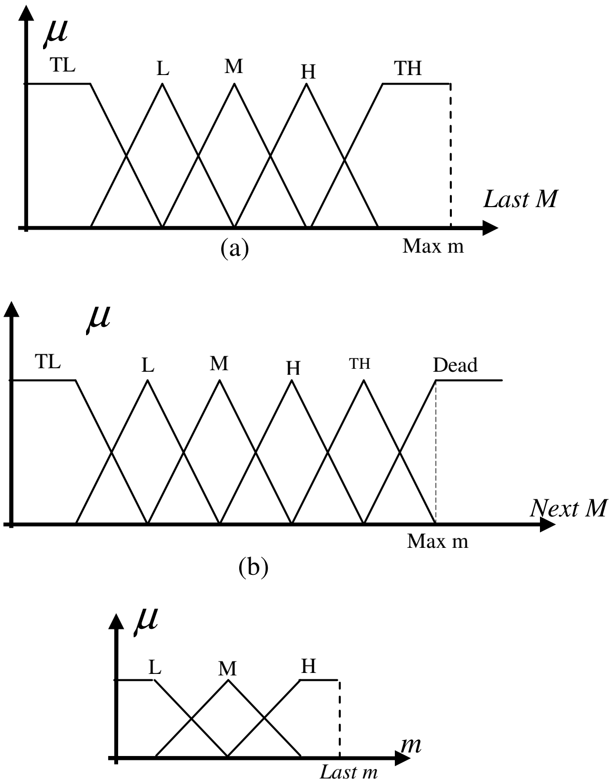
When its journey ends, the ant has to start a balancing operation between the overloaded (Max) and underloaded (Min) elements gathered during its roaming. In this stage, the number of elements in the Max-list and Min-list is close together (because of ant level load balancing). To achieve load balancing, the ant recommends underloaded nodes to the overloaded nodes and vice versa. In this way, the amount of load is dispersed equally among underloaded and overloaded nodes.

After load balancing, the ant should reinitiate itself to begin a new itineration. One of the fields that must be reinitiated is the ant's step counts. However, as stated in previous sections, the ant's step counts (m) must be commensurate to system conditions [6]. Therefore, if most of the visited nodes were underloaded or in equilibrium, the ant should prolong its wandering steps i.e. decrease the load balancing frequency and vice versa. Doing this requires the ant's knowledge about the environment. This knowledge should be based on the number of overloaded, underloaded and equilibrium nodes visited during the last itineration.

Because of fuzzy logic power in the adaptation among several parameters in a problem [22] and the consideration of the step counts (m) as a linguistic variable, e.g. short, medium, long, it is rational to use fuzzy logic for determining the next itineration step counts.

Actually, this is an adaptive fuzzy controller which determines the next itineration step counts (NextM, for short) based on the number of overloaded, underloaded and equilibrium nodes visited, along with the step counts during the last itineration (LastM). In other words, the number of overloaded, underloaded and equilibrium nodes encountered during the LastM indicate the recent condition of the environment, while the LastM, itself, reports the lifetime history of the ant.

The membership functions of the fuzzy sets are shown in Figure 2.



(c)

Figure 2: Membership functions of fuzzy sets
a) Last m, b) Next m, c) count of Max, Min and Average nodes visited

Where TL, L, M, H, TH shows Too Low, Low, Medium, High and Too High in Figure 2.a, 2.b and L, M, H indicates Low, Medium and High in Figure 2.c.

This fuzzy system can be displayed as a relation and a corresponding function as follows:

$$R_B : MaxCount<l,m,h>*MinCount<l,mh> \\ *Avg<l,m,h>\rightarrow NextM<TL,L,M, \\ H,TH,Dead> \qquad (2)$$

$$f(x) = \frac{\sum_{r=1}^{135} \bar{y}^r * \prod_{i=1}^{4} \mu_{B_i^r}(x_i)}{\sum_{r=1}^{135} \prod_{i=1}^{4} \mu_{B_i^r}(x_i)} \qquad (3)$$

Where x_i shows the input data into the system, $\bar{y}^r$ is the centre of the specific membership function declared in rule r. $\mu_{B_i^r}(x_i)$ indicates the membership value of the i th input in membership functions of the r th rule. In this inference system, we also have 4 inputs and 135 rules defined, as stated in (3).

In this system, a large number of underloaded and, especially, equilibrium elements indicate equilibrium states. Consequently, the NextM should be prolonged, thus lowering the load balancing frequency. One can say that, if an ant's step counts extend to extreme values, its effect tends to be zero. Based on this premise, we can conclude that an ant with overly long step counts does not have any influence on the system balance. Rather, the ant imposes its communication overhead on the system. In this situation, the ant must commit suicide. This is the last ring of the echo system. Therefore, if the *NextM* is fired in the *"Dead"* membership function, the ant does not start any new itineration.

Below is a diagram exhibiting the ant's behaviour in different environmental conditions. Figure 3.a shows the relation between the *LastM* and the amount of overloaded nodes visited, while Figure 3.b illustrates the relation between the *Last*M and the number of equilibrium nodes visited.

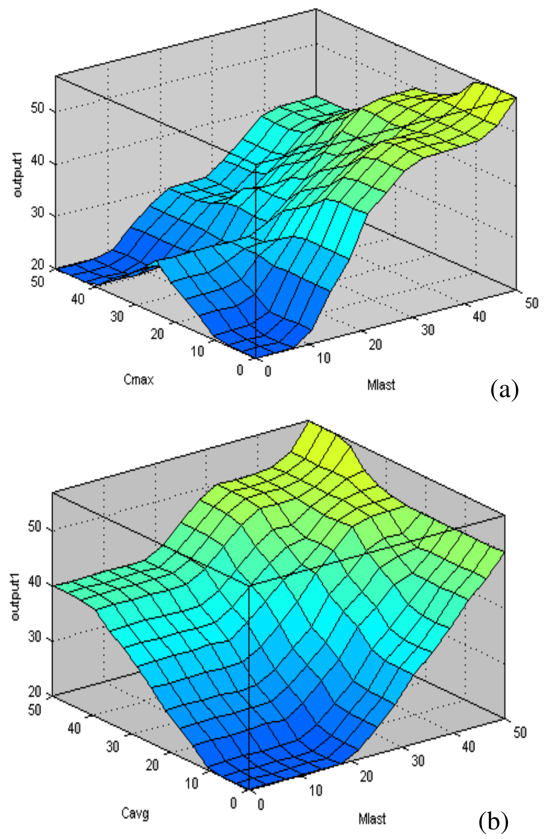


Figure 3: Schematic view of adaptive determining of next itineration step counts. a) LastM –MaxCount–output, b)LastM – avgCount–output

5 Performance valuations

In this section, several common statistics are investigated, which show the performance of the mechanism.

5.1 Efficiency

To prove that the new mechanism increases efficiency, it should be compared with the mechanism described in [4]. First, we introduce some of the most important criteria in load balancing:

Let P be the number of agents and Wpk where (p: 1, 2... P) is the workload of the agent p at step k. The average workload is:

$$\overline{W_k} = \frac{\sum_{p=1}^{P} W_{pk}}{P} \qquad (4)$$

The mean square deviation of Wpk, describing the load balancing level of the system, is defined as:

$$L_k = \sqrt{\frac{\sum_{p=1}^{P} (\overline{W_k} - W_{pk})^2}{P}} \qquad (5)$$

Finally, The system load balancing efficiency (e) is defined as:

$$e_k = \frac{L_0 - L_k}{C_k} \quad (6)$$

Where ek means efficiency at step k and Ck is the total number of agent connections that have achieved a load balancing level Lk. To compare the efficiency of these two mechanisms, we should consider $e_{k_{new}} / e_{k_{Trad}}$.

As L0 indicates the load balancing level at the beginning of the load balancing process and is also equal in both new and seminal mechanisms, we shall discuss the value of Lk. For the sake of simplicity, assume that every node gets to $\overline{W}_k$ after the balancing process and no longer requires balancing, i.e.

$$\overline{W}_k - W_{pk} = 0 \quad (7)$$

On the other hand, after the k stage, if the memory space considered for overloaded and underloaded elements is equal to 'a' (a>2), then we have ka elements balanced:

$$L_{k_{new}} = \sqrt{\frac{\sum_{p=1}^{p-ka} (\overline{W}_k - W_{pk})^2}{P}} \quad (8)$$

While in the seminal approach we have:

$$L_{k_{Trad}} = \sqrt{\frac{\sum_{p=1}^{p-2k} (\overline{W}_k - W_{pk})^2}{P}} \quad (9)$$

If we suppose that a>2, we can conclude:

$$P - 2k > P - ka \quad (10)$$

After the k stages, the difference in the balanced nodes in these two mechanisms is:

$$P - 2a - P + ka = k(a-2) \quad (11)$$

Then:

$$L_{k_{Trad}} = \sqrt{\frac{\sum_{p=1}^{p-ka} (\overline{W}_k - W_{pk})^2}{P} + \frac{\sum_{p=ka}^{p-2k} (\overline{W}_k - W_{pk})^2}{P}} \quad (12)$$

$$L_{k_{new}} = \sqrt{\frac{\sum_{p=1}^{p-ka} (\overline{W}_k - W_{pk})^2}{P}} \quad (13)$$

$$L_{k_{Trad}} > L_{k_{new}} \rightarrow \frac{L_{k_{new}}}{L_{k_{Trad}}} < 1 \quad (14)$$

With respect to (14), we have:

$$\frac{e_{k_{new}}}{e_{k_{Trad}}} = \frac{2(L_0 - L_{k_{new}})}{L_0 - L_{k_{Trad}}} \Rightarrow \frac{e_{k_{new}}}{e_{k_{Trad}}} > 2 \quad (15)$$

One of the most important parameters in the efficiency of the new mechanism is the ant's memory space (a). In an extreme case, if a=2, then the mechanism resembles the seminal one, with half steps (S), i.e.

$$S_{new} = 1/2 * S_{Trad} \quad (16)$$

Consider that memory space (a) is effective if and only if it can be filled during the ant's wandering steps. Therefore, if a increases, then the amount of steps (S) must increase accordingly to prevent performance degradation. This means that:

If $a \rightarrow \infty$ then $S \rightarrow \infty$ (17)

Increasing S causes a decrease in load balancing frequency and consequently an increase in convergence time.

Overly long trips also lead to many reserved nodes. At the same time, there may be other roaming ants looking for free, unbalanced nodes. On the other hand, expanding the ant's memory leads to occupying too much bandwidth as well as increasing processing time. Actually, there is a trade-off between the step counts (S) and the memory allocated to each ant (a).

If a<<S, then the memory allocated expires rapidly and the ant is compelled to generate new ants. This explodes the ant population, subsequently augments their communication and the remaining pheromone and finally leads to an increase in time. However, as the probability of balancing every node more than once rises, the load balancing level falls.
On the other side, if a→S, then the probability of creating new ants lessens. Subsequently, the ant's population is reduced. Cutting down on the ant population results in faster speed, diminished communication and the pheromone left by the ants. The final result, however, is not satisfactory (final load balancing level is high). Due to the reasons discussed and with respect to several experiments shown in Figures 4, 5 and Table 1, we deduce that, in order to satisfy the different parameters mentioned, it is better to set the allocated memory at about half of the step counts.

$$a \cong S/2 \quad (18)$$

Experiments achieved with a different memory size allocated, where S=15 initially, are reported here.

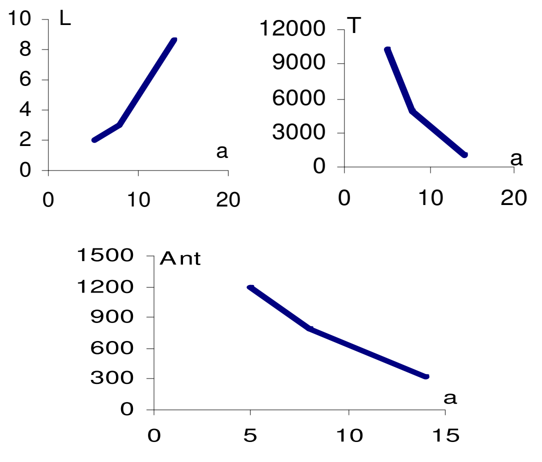


Figure 4: Relation between memory allocated to each ant (a) and a) load balancing level(L) b) time(T) base on millisecond c) Ant no (Ant), where the Ant initial Step Counts (S)=1

a	Time(ms)	Level	Ant No
5	10274	1.9455	1197
8	4906	3.0363	797
14	971	8.6015	325

Table 1: Relation between ants' memory size (a) and Ants with initial Step Counts (S=15).

5.2 Load balancing speed

Adaptively determining the step counts, actually, causes a differentiation in load balancing frequency over time. In other words, as time increases, the whole system approaches convergence and the load balancing frequency lessens, hence postponing the final convergence time. On the contrary, the new mechanism imposes less overhead as the system nears the balance state. In reality, in an environment such as the grid, attaining final convergence is impractical because of its inherent dynamism and vastness. However, if balancing occurs, it would not last long.

Figure 5 shows a schematic comparison for load balancing frequency between the new and the seminal mechanism.

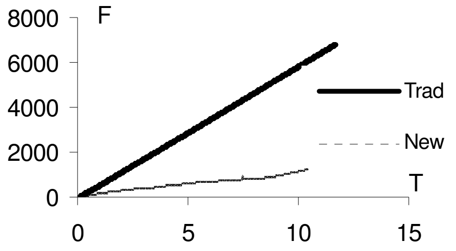


Figure 5: Comparison between the Seminal (Trad) and the new method's load balancing frequency (F).

5.3 Experimental results

Experiments are achieved according to the specifications of Simorgh mini-grid [25]. This mini-grid will include different clusters throughout Ferdowsi University. However, as this mini-grid is under construction now, we have simulated its behaviour. In this simulation, Agent system, Workload, and Resources are modelled as follows:

• Agents. Agents are mapped to a square grid. This simplification has been done in similar works [6, 13]. All of experiments described later include 400 agents.

• Workload. A workload value and corresponding distribution are used to characterize the system workload. The value is generated randomly in each agent.

• Resources. Resources are defined in the same way as workload.

The first experiment involves total network connections. In this experiment, as shown in Figure 6.a, ant communication in the new mechanism is drastically less than in the seminal approach. This is mainly because every time an ant wanders 'S' steps in the new method, it balances 'k' elements. In the traditional method, however, the ant wanders '2S+1' steps and then balances only two elements. Therefore, as seen below, with an equal initial step count (S=15), the ant in the new mechanism only goes through 2,000 stages to achieve final convergence, while in the traditional method, the ant passes 7,000 stages. Figure 6.b illustrates the comparison between a colony of ants using S=15 and a memory size=7. This figure elucidates that, in the new mechanism, the communication count goes flat. This occurs when the step counts enlarge and load balancing frequency decreases, i.e. in the last seconds.

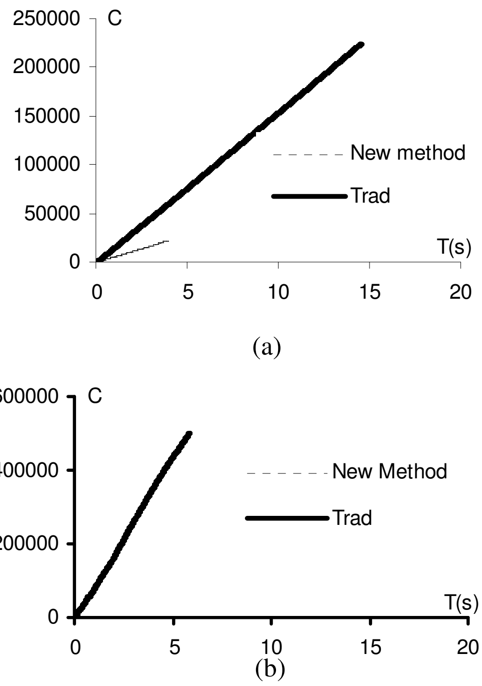


Figure 6: Comparing agent communications (C) between the new and seminal (Trad) method. Final results using. a) One ant S=15, a=7 b) a colony of ants, N=220, S=15, a=7

The second experiment focuses on the relation between load balancing levels and the number of dead ants. As illustrated in Figure 7, as the number of dead ants rises, the load balancing level declines, i.e. it approaches final convergence. This experiment is conducted with different initial ants. Repeating the experiment with a different number of initial ants proves that, deploying more ants would result in better balancing level.

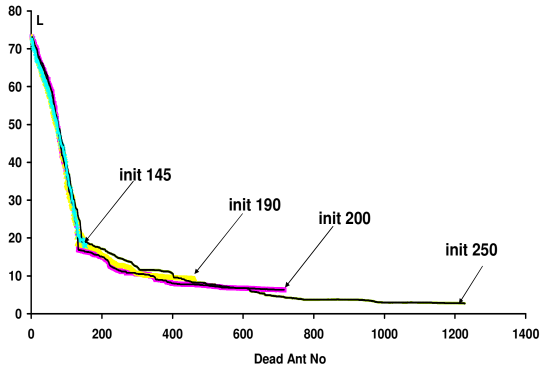


Figure 7: Impact number of created ants (DeadAntNo) on the load balancing level (L), the experiment achieved with different numbers of initial ants (init).

The third experiment concentrates on the correlation between an ant's step counts and the load balancing level. The average step counts of the swarm over time are used for measurement. As Figure 8 shows, the step count increases by approaching convergence. This results in delay to achieve final convergence.

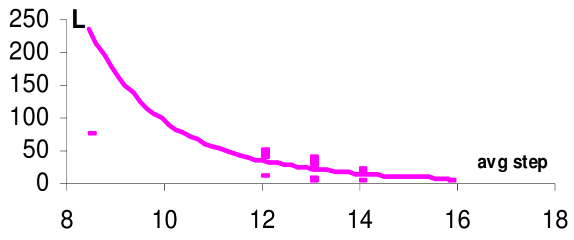


Figure 8: Relation between average Step count (avgStep) and load balancing Level (L)

The fourth experiment indicates the effect of proposed load balancing method on the final job distribution. As understood from Figure 9, ant level load balancing produces a better convergence.

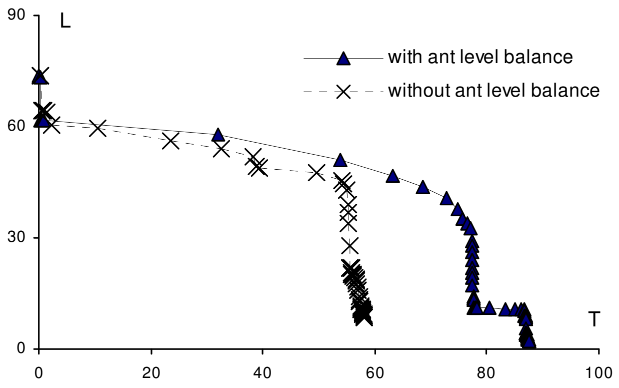


Figure 9: Comparison between effects of using ant level load balancing on final balancing level (L) during the time (T).

It is clear that this load balancing method cannot be achieved without any cost. As illustrated in Figure 9, although the proposed method results are better than previous ones, it consumes more time. We must acknowledge that the new method enables the ant to obtain global information even while moving locally. Furthermore, the validity of the exchanged information is guaranteed in contrast to using the pheromone, which is not, even with evaporation.

The fifth experiment presents the efficiency of the new method in comparison with the seminal approach. Figure 10 illustrates that the new method, with different initial steps and different memory allocated, is more efficient than the seminal one.

On the other hand, comparing the new method's efficiencies, with different initial step counts(S) and different memory allocated shows the effect of the trade-off in determining the memory allocated to each ant (a). In this case, if the memory allocated is high, e.g. a=10, then the probability of creating new ants decreases. So, the probability of visiting a node by different ants is decreased which causes a fall in efficiency. In the other way, as mentioned earlier, low values for memory allocated (a), e.g. a=5, increase the ant population and consequently their interconnections (Ck). This again results in decreasing the final efficiency in regard to (6).

Consider that stages do not have a completely standard meaning in our method. Thus, we think of periods of time as stages (k).

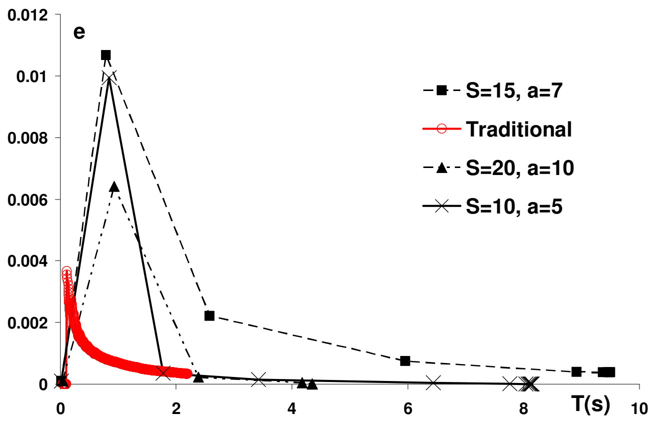


Figure 10: Efficiency (e) comparison between the traditional and the new method with different step counts and memory allocated, during the time (T).

6 Conclusion

As described in the previous sections, equalizing the load of all available resources is one of the most important issues in the grid. In this way, with respect to grid specifications, an echo system of autonomous, rational and adaptive ants is proposed to meet the challenges of load balancing. There are great differences between the proposed mechanism and similar mechanisms which deploy ant colony optimization. We believe that ant level load balancing is the most important difference.

In future work, we plan to extend the applications of ant level load balancing in addition to implementing this mechanism in a more realistic environment. Promoting ant intelligence and adaptation, establishing billing contracts among resources as they exchange customer loads, as well as overcoming security concerns are other future work.

References

[1] F. Berman, Anthony J.G. Hey, Geoffrey C. Fox (2003) *Grid Computing Making the Global Infrastructure a Reality* WILEY SERIES IN COMMUNICATIONS NETWORKING & DISTRIBUTED SYSTEMS.

[2] J. Cao, S. A. Jarvis, S. Saini, D. J. Kerbyson, and G. R. Nudd (2002) *ARMS: an Agent-based Resource Management System for Grid Computing*, Scientific

Programming, Special Issue on Grid Computing Vol. 10, No. 2 pp. 135-148.

[3] M. Baker, R. Buyya, and D. Laforenza (2002) *Grids and Grid Technologies for Wide-area Distributed Computing*, Software: Practice and Experience Vol. 32, No. 15 pp. 1437-1466.

[4] J. Cao, D. J. Kerbyson, G. R. Nudd (2001) *Performance evaluation of an agent-based resource management infrastructure for grid computing* in Proc. 1st IEEE Int. Symp. on Cluster Computing and the Grid, pp. 311-318.

[5] J. Cao and F. Zimmermann (2004) *Queue Scheduling and Advance Reservations with COSY* in Proc. of 18th IEEE Int. Parallel and Distributed Processing Symp pp. 120-128.

[6] J. Cao (2004) *Self-Organizing Agents for Grid Load Balancing* Proc. of the 5th IEEE/ACM Int. Workshop on Grid Computing, pp.168-176.

[7] A. Y. Zomaya, and Y. The (2001) *Observations on using genetic algorithms for dynamic load-balancing*, IEEE Trans. on Parallel and Distributed Systems, Vol. 12, No. 9, pp. 899-911.

[8] O. Remien, J. Kramer (1992) *Methodical analysis of adaptive load sharing algorithms*, IEEE Trans. on Parallel and Distributed Systems, Vol. 3, No: 11, pp. 747-760.

[9] Bing Qi Chunhui Zhao (2007) Ant Algorithm Based Load Balancing for Network Sessions, *ICNC 2007*, 3th Int. Conference on Natural Computation, pp. 771-775.

[10] Y. Lan, T. Yu (1995) *A Dynamic Central Scheduler Load-Balancing Mechanism*, Proc. 14th IEEE Conf. on Computers and Communication, Tokyo, Japan, pp. 734-740.

[11] H.C. Lin, C.S. Raghavendra (1992) *A Dynamic Load-Balancing Policy with a Central Job Dispatcher (LBC),* IEEE Transaction on Software Engineering, Vol. 18, No. 2, pp. 148-158.

[12] Z. Zeng, B. Veeravalli (2004) *Rate-Based and Queue-Based Dynamic Load Balancing Algorithms in Distributed Systems*, *Proc.* 10th IEEE Int. Conf. on Parallel and Distributed Systems, pp. 349- 356.

[13] M. Amini, H. Deldari (2006) *Grid Load Balancing Using an Echo System of Ants*, Proc. Of 24th IASTED Int. Conf, Innsbruck, pp. 47–52.

[14] E. Bonabeau, M. Dorigo, G. Theraulaz (1999) *Swarm Intelligence: from natural to artificial systems*, Oxford University Press, pp: 75-98.

[15] M. T. Islam, P. Thulasiraman, R. K. Thulasiram (2003) *A Parallel Ant Colony Optimization Algorithm for All-Pair Routing in MANETs*, Proc. 3th Int. Symp., Parallel and Distributed Processing, pp. 259-270.

[16] Siriluck Lorpunmanee, Mohd Noor Sap, Abdul Hanan Abdullah, and Chai Chompoo-inwai (2007) *An Ant Colony Optimization for Dynamic Job Scheduling in Grid Environment*, Int. Journal of Computer and Information Science and Engineering Volume 1 Number 4, pp. 207-214.

[17] J. Deneubourg, S. Goss, N. Franks (1990) *The dynamics of collective sorting robot-like ants and ant-like robots*, Proc. of the 1st Int. Conf. on Simulation of Adaptive Behavior, pp. 356-363.

[18] Ö. Babaoglu, H. Meling, A. Montresor (2002) *Anthill: A Framework for the Development of Agent-Based Peer-to-Peer Systems*, in Proc. of 22th IEEE Int. Conf. on Distributed Computing Systems, Vienna, Austria, pp. 15-22.

[19] J. Liu, X. Jin, and Y. Wang (2005) *Agent-Based Load Balancing on Homogeneous Minigrids: Macroscopic Modeling and Characterization*, IEEE TRANS. ON PARALLEL AND DISTRIBUTED SYSTEMS, VOL 16, NO 7, pp.586-598.

[20] A. Montresor, H. Meling, and Ö. Babaoglu (2002), Messor: *Load-Balancing through a Swarm of Autonomous Agents*, in Proc. of 1st ACM Int. Joint Conf. on Autonomous Agents and Multi-Agent Systems, Bologna, Italy, pp.112-120.

[21] A. Shaout, P. McAuliffe (1998) *Job scheduling using fuzzy load balancing in distributed system*, ELECTRONICS LETTERS, Vol. 34, No. 20, pp. 56-62.

[22] Hai Zhuge, Jie Liu (2004) *A fuzzy collaborative assessment approach for Knowledge Grid*, Int. Journal of Future Generation Computer Systems, Vol. 2, No 20, pp. 101-111.

[23] M. Dorigo, L. Maria (1997) *Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem*, Transactions ON EVOLUTIONARY COMPUTATION, VOL. 1, NO. 1, pp. 53-66.

[24] M. Dorigo, G. Carol (1999) *Ant Colony Optimization: A New Meta-Heuristic*, proc. Of 3th Fuzzy Sets and Systems, pp. 21–29.

[25] http://profsite.um.ac.ir/~hpcc

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 800 staff, has 600 researchers, about 250 of whom are postgraduates, nearly 400 of whom have doctorates (Ph.D.), and around 200 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 **Slove**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.

From 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.

Part of the Institute was 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 was developed at a particularly historical moment, characterized by the process of state reorganisation, privatisation and private initiative. The national Technology Park is a shareholding company hosting an independent venture-capital institution.

The promoters and operational entities of the project are the Republic of Slovenia, Ministry of Higher Education, 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 the Economy, 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 1 251 93 85
WWW: http://www.ijs.si
E-mail: matjaz.gams@ijs.si
Public relations: Polona Strnad

INFORMATICA

AN INTERNATIONAL JOURNAL OF COMPUTING AND INFORMATICS

INVITATION, COOPERATION

Submissions and Refereeing

Please submit an email with the manuscript to one of the editors from the Editorial Board or to the Managing Editor. At least two referees outside the author's country will examine it, and they are invited to make as many remarks as possible from typing errors to global philosophical disagreements. The chosen editor will send the author the obtained reviews. If the paper is accepted, the editor will also send an email to the managing editor. The executive board will inform the author that the paper has been accepted, and the author will send the paper to the managing editor. The paper will be published within one year of receipt of email with the text in Informatica MS Word format or Informatica LaTeX format and figures in .eps format. Style and examples of papers can be obtained from http://www.informatica.si. Opinions, news, calls for conferences, calls for papers, etc. should be sent directly to the managing editor.

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, 1000 Ljubljana, Slovenia. E-mail: drago.torkar@ijs.si

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:

http://www.informatica.si/

Referees:

Informatica

An International Journal of Computing and Informatics

Web edition of Informatica may be accessed at: http://www.informatica.si.

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 2008 (Volume 32) is
– 60 EUR for institutions,
– 30 EUR for individuals, and
– 15 EUR 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.
Printing: Dikplast Kregar Ivan s.p., Kotna ulica 5, 3000 Celje.

Orders may be placed by email (drago.torkar@ijs.si), telephone (+386 1 477 3900) or fax (+386 1 251 93 85). The payment should be made to our bank account no.: 02083-0013014662 at NLB d.d., 1520 Ljubljana, Trg republike 2, Slovenija, IBAN no.: SI56020830013014662, SWIFT Code: LJBASI2X.

Informatica is published by Slovene Society Informatika (president Niko Schlamberger) 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: Citeseer, COBISS, Compendex, Computer & Information Systems Abstracts, Computer Database, Computer Science Index, Current Mathematical Publications, DBLP Computer Science Bibliography, Directory of Open Access Journals, InfoTrac OneFile, Inspec, Linguistic and Language Behaviour Abstracts, Mathematical Reviews, MatSciNet, MatSci on SilverPlatter, Scopus, Zentralblatt Math

The issuing of the Informatica journal is financially supported by the Ministry of Higher Education, Science and Technology, Trg OF 13, 1000 Ljubljana, Slovenia.

Volume 32 Number 3 October 2008 **ISSN 0350-5596**

Informatica

An International Journal of Computing and Informatics

Informatica **32** (2008) Number 3, pp. 229–337