Heuristics for Optimization of LED Spatial Light Distribution Model

David Kaljun and Darja Rupnik Poklukar
Faculty of Mechanical Engineering, University of Ljubljana, Aškerčeva 6, 1000 Ljubljana, Slovenia
E-mail: david.kaljun@fs.uni-lj.si

Janez Žerovnik
Faculty of Mechanical Engineering, University of Ljubljana, Aškerčeva 6, 1000 Ljubljana, Slovenia and
Institute of Mathematics, Physics and Mechanics, Jadranska 19, Ljubljana, Slovenia
E-mail: janez.zerovnik@fs.uni-lj.si

Keywords: local search, iterative improvement, steepest descent, genetic algorithm, Wilcoxon test, least squares approximation

Received: December 1, 2014

Recent development of LED technology enabled production of lighting systems with nearly arbitrary light distributions. A nontrivial engineering task is to design a lighting system or a combination of luminaries for a given target light distribution. Here we use heuristics for solving a problem related to this engineering problem, restricted to symmetrical distributions. A genetic algorithm and several versions of local search heuristics are used. It is shown that practically useful approximations can be achieved with majority of the algorithms. Statistical tests are performed to compare various combinations of parameters of genetic algorithms, and the overall results of various heuristics on a realistic dataset.

Povzetek: Napredek tehnologije LED je omogočil izdelavo osvetljevalnih sistemov s skoraj poljubno porazdelitvijo svetlobe. Netrivialna inženirska naloga je, kako načrtovati osvetljevalni sistem ali kombinacijo svetilk za dano ciljno porazdelitev svetlobe. V sestavku predstavljamo uporabo hevrističnih algoritmov za reševanje te naloge, kjer predpostavljamo, da je porazdelitev svetlobe osno simetrična. Izkaže se, da lahko dobimo praktično uporabne rešitve z algoritmi za lokalno optimizacijo, z genetskimi algoritmi in s hibridnimi algoritmi, ki povezujejo obe ideji. Za izbiro parametrov genetskih algoritmov in za primerjavo različnih algoritmov na izbranem vzorcu realnih podatkov so uporabljeni statistični testi.

1 Introduction

Even the most simply stated optimization problems such as the traveling salesman problem are known to be NP-hard, which roughly speaking means that there is no practical optimization algorithm provided the famous P≠NP conjecture is correct [26]. From practical point of view, knowing that the problem is computationally intractable implies that we may use heuristic approaches. It is well known that best results are obtained when a special heuristics is designed and tuned for each particular problem. This means that the heuristics should be based on considerations of the particular problem and perhaps also on properties of the most likely instances. On the other hand, it is useful to work within a framework of some (one or more) metaheuristcs which can be seen as a general strategies to attack an optimization problem. Metaheuristics in contrast to heuristics often make fewer assumptions about the optimization problem being solved, and so they may be usable for a variety of problems, while heuristics are usually designed for particular problem or even particular type of problem instances. Compared to optimization algorithms, metaheuristics do not guarantee that a globally optimal solution can be found on some class of problems. We say that the heuristics search for so called near optimal solutions because in general we also have no approximation guarantee. Several books and survey papers have been published on the subject, for example [25].

Most studies on metaheuristics are experimental, describing empirical results based on computer experiments with the algorithms. As experiments provide only a sample that may in addition be biased for a number of reasons, it is often hard to draw any firm conclusions from the experimental results, even when statistical analysis is applied (see, c.f. [4] and the references there). Some theoretical results are also available, often proving convergence of a particular algorithm or even only showing the possibility of finding the global optimum.

Perhaps the most natural and conceptually simple metaheuristics is local search. In the search space of feasible solutions that is usually regarded as a "landscape", the solutions with extremal values of the goal functions are to be found. In order to speak about local search on the landscape, a topology is introduced, usually via definition of a neighborhood structure. It defines which feasible solutions can be obtained in "one step" from a given feasible solution. It is essential that the operation is computationally

cheap and that the new value of the goal function is provided. There are two basic variants of the local search, iterative improvement and best neighbor (or steepest descent). As the names indicate, starting from initial feasible solution, iterative improvement generates a random neighbor, and moves to the new solution based on the difference in goal function. The procedure stops when there has been no improvement for sufficiently long time. On the other hand, best neighbor heuristics considers all neighbors and moves to the new solution with best value of the goal function. If there is no better neighbor, the current solution is clearly a local optima. Note that given a particular optimization problem, often many different neighborhood structures can be defined giving rise to different local search heuristics. Recently, there has been some work on the heuristics that use and switch among several neighborhoods [21].

In fact, most metaheuristics can be seen as variations or improvement of the local search [1]. Examples of popular metaheuristics that can be seen as variations of local search include iterated local search, simulated annealing [17], threshold accepting [7], tabu search [11], variable neighborhood search [21], and GRASP (Greedy Randomized Adaptive Search Procedure) [8]. The other type of search strategy has a learning component added to the search, aiming to improve the obvious drawback of the local search, complete lack of memory. (An exception is the tabu search that successfully introduces a short time memory.) Metaheuristics motivated by idea of learning from past searches include ant colony optimization [6, 28, 10, 9, 19], evolutionary computation [3] and its special case, genetic algorithms, to name just a few. It is however a good question in each particular case whether learning does indeed mean an improvement [29], namely a successful heuristic search must have both enough intensification and diversification.

Genetic algorithms (GA) are optimization and search techniques based on the natural evolution principles. The basic idea is to allow a population composed of many individuals to evolve under specified selection rules to a point where some of the population individuals reach or at least get close to the optimal solution. The method was developed by John Holland, and popularized by one of his students, David Goldberg, who was able to solve a difficult problem involving the control of gas-pipeline transmission for his dissertation. Since the early days of GA, many versions of evolutionary based algorithms have been tried with varying degrees of success. Nevertheless there are some advantages of GA worth noticing [12, 23]. GA is able to work with continuous or discrete variables, does not require derivative information, it simultaneously searches from a wide sampling of the cost surface, deals with a large number of variables, is well suited for parallel computers, optimizes variables with extremely complex cost surfaces (they can jump out of a local minimum), provides a list of optimum variables not just a single solution and works well with numerically generated data, experimental data, or analytical functions.

In this paper, a comprehensive experimental study of several heuristics on an industrial problem is carried out. It extends and upgrades previous published work on the subject, in particular by introducing a statistically based comparison of the algorithms. The results of algorithms are statistically tested in order to determine significant differences between them. Another extension of previous work is the genetic algorithm parameter tunning, presented bellow. Previous related work is the following. The suitability of the model and practical applicability have been shown in [14]. Attempting to improve and speed up the optimization, different metaheuristics have been implemented and compared. The conference paper [13] reports results of a comparison of local search with a naive genetic algorithm. A hybrid genetic algorithm was proposed in [15].

Here we implement and run two versions of genetic algorithm, a standard genetic algorithm (SGA) and a hybrid genetic algorithm (HGA) where we infuse a short local search as an evolution rule in hope to enhance the population. As the initial experiment was run on various computers, and consequently the results on various computers slightly differed because of different environments and in particular different random generators. The new experiment therefore repeated the complete experiment, this time on the same computer, a standard home PC with a Intel Core I7-4790K @ 4.4 GHz processor. The experiment was run in parallel on 6 threads. In addition, part of the code was rewritten to make it more machine independent. Furthermore, statistical tests on the experimental results were applied thus providing ground for tuning the parameters of genetic algorithms and for comparison of various algorithms' performance on the dataset considered.

The rest of the paper is organized as follows. In the next section we provide background of the engineering application. Section 3 provides the analytical model and the optimization problem that is addressed. In Section 4, overview of the experimental study is given. Details of local search heuristics and genetic algorithms used are given in Sections 5 and 6. Section 7 elaborates tuning of parameters for the genetic algorithms. Main experiment, comparison of local search, standard and hybrid genetic algorithms is presented in Section 8. The paper ends with a summary of conclusions and ideas for future work, Section 9.

2 Motivation – the Engineering Problem

The mass production of high power - high efficacy white Light Emitting Diodes (LEDs), introduced a revolution in the world of illumination. The LEDs at the basics enable lower energy consumption, never before seen design freedoms and of course endless possibilities on the design of optics systems. The latter in turn enables the optics designer to build a lighting system that delivers the light to the environment in a fully controlled fashion. The many possible designs lead to new problems of choosing the optimal or at least near optimal design depending on possibly

different goals such as optimization of energy consumption, production cost, and, last but not least, the light pollution of the environment. Nevertheless the primary goal or challenge of every luminaire design process is to design a luminaire with an efficient light engine. A light engine consists of a source, which are LEDs, and the appropriate secondary optics. The choice of the secondary optics is the key in developing a good system while working with LEDs. For designing such a system nowadays technology provides two options. The first option is to have the know-how and the resources to design a specific lens to accomplish the task. However, the cost of resources coupled with the development and production of optical elements may be enormous. Therefore a lot of manufactures are using the second option, that is to use ready made of the shelf lenses. These lenses are produced by several specialized companies in the world that offer different types of lenses for all of the major brands of LEDs. The trick here is to choose the best combination of lenses to get the most efficient system. The usual current practice in development process is a trial and error procedure, where the developer chooses a combination of lenses, and then simulates the system via Monte Carlo ray-tracing methods. The success heavily depends on the engineers' intuition and experience but also needs sizable computation resources for checking the proposed design by simulation. In contrast to that, we believe that using analytical models and optimization tools may speed up the design and also at the same time possibly improve the quality of solutions. The first step towards this ambitious goal is to investigate an analytical model and its use for representing single ready made lenses. For this purpose we adopt an analytical model presented by Moreno and Sun [22] and use heuristic methods based on this model to provide good approximations.

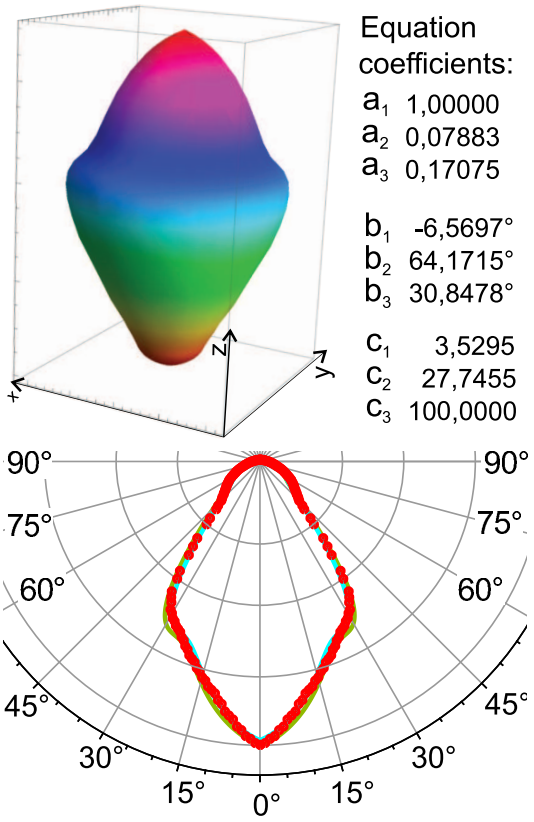

Figure 1: Fitting results on the C13353 lens with the 3D represenation.

3 Analytical Model and Problem Definition

With so many different LED's that have different beam patterns and many different secondary optics that can be placed over these LED's to control the light distribution, finding the right combination of a LED - lens combo is presumably a very complicated and challenging task. Consequently, providing a general analytical model for all of them is also likely to be a very challenging research problem. Here we therefore restrict attention to LED-lens combinations that have symmetrical spatial light distributions. In other words, the cross section of the surface which represents the spatial distribution with a section plain that is coincident with the vertical axis of the given coordinate system is alike at every azimuthal angle of offset. This yields an analytical model in two dimensions, so it describes a curve rather a surface. To produce the desired surface, we just revolve the given curve around the central vertical axis with the full azimuthal angle of 360°.

In [14], a normalizing parameter I_{max} is introduced in addition to the parameters of the original model [22] as this simplifies (unifies) the range intervals of the other three parameters: $a = [0, 1]$, $b = [0, 90]$ and $c = [0, 100]$, for all test lenses. The model used is based on the expression

$$I\left(\Phi; \mathbf{a}, \mathbf{b}, \mathbf{c}\right) = I_{max} \sum_{k=1}^{K} a_k * \cos(\Phi - b_k)^{c_k} \quad (1)$$

Assume that we have measured values $I_m(\Phi_i)$ at angles Φ_i, $i = 1, 2, \ldots, N$. The goodness of fit is, as usual, defined to be minimizing the root mean square error (RMS), or, formally [22, 24]:

$$\mathrm{RMS}\left(\mathbf{a}, \mathbf{b}, \mathbf{c}\right) = \sqrt{\frac{1}{N} \sum_{i=1}^{N} \left[I_m(\Phi_i) - I(\Phi_i, \mathbf{a}, \mathbf{b}, \mathbf{c})\right]^2} \quad (2)$$

For a sufficiently accurate fit, the RMS value must be less than 5% [22, 24]. On the other hand, current standards and technology allow up to 2% noise in the measured data. Therefore, the target results of the fitting algorithms are at less than 5% RMS error, but at the same time there is no practical need for a solution with less than 1% or 2% RMS error.

We will assume that all data is written in form of vectors v= (polar angle [Φ], intensity [I]). In reality, measured

photometric data from the lens manufacturers is available in one of the two standard coded formats. These are the IESNA photometric digital format *.ies [27] used primarily in the USA and the European format EULUMDAT *.ldt [2]. The data in the two standard formats can easily be converted into a list of vectors. In addition, due to the parameter I_{max} each dataset will be normalized during the preprocessing so that in each instance the maximal intensity of the vectors will be 1, and the normalizing value I_{max} is given as additional input value to the algorithms.

The problem can formally be written as:

INPUT: I_{max} and a list of vectors v= (polar angle [Φ], intensity [I])
TASK: Find parameters (a_1, b_1, c_1, a_2, b_2, c_2, a_3, b_3, c_3) that minimize the RMS error (2).

4 Overview of the Experimental Study

Although the minimization problem defined above is conceptually simple, it is on the other hand likely to be computational hard. In other words, it is a min square error approximation of a function for which no analytical solution is known.

The experiment was set-up to test the algorithms performance on different real life LED-lens combinations.

We have chosen a set of real available lenses to be approximated. The set was taken from the online catalogue of one of the biggest and most present manufacturer in the world Ledil Oy Finland [18]. The selection from the broad spectrum of lenses in the catalogue was based on the decision that the used LED is of the XP-E product line from the manufacturer Cree [5]. And the demand that the lenses have a symmetric spatial light distribution. We have preserved the lens product codes from the catalog, so the reader can find the lens by searching the catalog for the code from the first column in tables below, c.f. Table 1.

All of the chosen lenses were approximated with all algorithms. To ensure that algorithms' results could be compared the target error was set to 0% and the runtime was defined in terms of basic steps that is defined as a generation of a feasible solutions in the local search and an adequate operation for genetic algorithms. This implies that the wall clock runtime was also roughly the same for all algorithms. Details are given below.

In the experiment and in the study, we address the optimization problem as a discrete optimization problem. Natural questions that may be asked here is why use heuristics and why discrete optimization heuristics on a continuous optimization problem. First, application of an approximation method is justified because there is no analytical solution for best approximation of this type of functions. Moreover, in order to apply continuous optimization methods such as the Newton method, usually we would need a good approximation in order to assure convergence. Therefore a method for finding good starting solution before running fine approximation based on continuous optimization methods is needed. However, in view of the at least 2% noise in the data, these starting solutions may in many cases already be of sufficient quality! Nevertheless, it may be of interest to compare the two approaches and their combination in future work, although it is not of practical interest for the engineering problem regarded here.

When considering the optimization problem as a discrete problem, the values of parameters to be estimated will be $a_\star \in [0, 0.001, 0.002, \ldots, 1]$, $b_\star \in [-90, -89.9, -89.8, \ldots, 90]$, and $c_\star \in [0, 1, 2, \ldots, 100]$. Hence, the discrete search space here consists of $N_t = 1000^i * 1800^i * 100^i \sim 5,83 * 10^{24}$ tuples $t = (a_1, a_2, a_3, b_1, b_2, b_3, c_1, c_2, c_3)$.

In the experiments, all the heuristics were tested on all instances of the dataset, a long run and a short run. The long run is defined to be 4 million steps that are defined to be equivalent of one iteration of a basic local search heuristics, in other words it is the number of feasible solutions generated by the iterative improvement. The time for other heuristics is estimated to be comparable, and will be explained in detail later. Short runs are one million and two hundred thousand steps long and the long runs have four million steps. The long run CPU time per algorithm and lens was measured to be 16 minutes on the processor Intel Core I7-4790K @ 4.4 GHz and 16 GB of RAM. The code is not fully optimized. The overall runtime of the experiment was substantially lowered by use of parallelization. We ran the experiment on 6 of the 8 available CPU threads.

5 Local Search Heuristics

First we discuss the specific local search type heuristics. As the original problem is a continuous optimization problem, compared to discrete optimization, there are even more possibilities to define a neighborhood for the local search based heuristics. In fact, the neighborhoods we use can be seen as variable neighborhoods though they are all similar. Below we define two neighborhoods that were implemented.

We have started our experiments with two basic local search algorithms, steepest descent (SD) and iterative improvement (IF), where in both cases the neighborhoods were defined in the same way. We call this neighborhood fixed step size neighborhood. The third local search algorithm (IR) is iterative improvement using a second type of neighborhood with random step size. Roughly speaking, given a step size and direction as before, we randomly make a step in the direction chosen and the step is at most as long as in the fixed size neighborhood search. Of course, there may be other neighborhoods that would be worth consideration. The main reason for not extending the selection of neighborhoods is simply the fact that they already gave us results of sufficient quality. The local search type heuris-

tics used here are explained in more detail below.

5.1 Steepest Descent (SD)

The *steepest descent* (SD) algorithm begins with the initialization of the initial function parameter values that are $a_1 = a_2 = a_3 = 0.5$, $b_1 = b_2 = b_3 = 0$, and $c_1 = c_2 = c_3 = 1$. Next it initializes the search step values which are for $da = 0.01$, for $db = 1$ and for $dc = \frac{I_{max}}{10}$ giving the 512 neighbors of the initial solution: $(a_1 \pm da, b_1 \pm db, c_1 \pm dc, a_2 \pm da, b_2 \pm db, c_2 \pm dc, a_3 \pm da, b_3 \pm db, c_3 \pm dc)$. If there are several neighbors with better RMS value, the search moves to the neighbor with minimal RMS value (if there are more minimal neighbors, one of them is chosen, all with the same probability). If none of the 512 is better than the current solution, a new set of neighboring solutions is generated, this time with a step size of $d_{n+1} = d_n + d_0$. This is repeated until $n = 10$. If there still is no better solution the search stops, the initial step value is multiplied by 0.9 and the search resumes from the current solution with a smaller initial step. The algorithm stops when the number of generated solutions reaches T_{max}.

5.2 Iterative Improvement – Fixed Neighborhood (IF)

The *iterative improvement with fixed neighborhood* (IF) algorithm initializes the same neighborhood as SD. Instead of considering all 512 neighbors, the algorithm generates a neighbor randomly, and immediately moves to that neighbor if its RMS value is better than the current RMS value. If no better neighbor is found after 1000 trials, it is assumed that no better neighbor exists. As above, the algorithm changes the size of the step value and continues the search in the same manner as SD algorithm does. The algorithm stops when the number of generated solutions reaches T_{max}.

5.3 Iterative Improvement – Variable Neighborhood (IR)

The *iterative improvement with a variable neighborhood* (IR) algorithm begins as the previous two algorithms. It initializes the same initial function parameter values but a different neighborhood which has the search step value within a range, rather than a static fixed value. The ranges are for $da_1 = da_2 = da_3 = \{-0.1, -0.099, -0.098, \ldots, 0.1\}$, for $db_1 = db_2 = db_3 = \{-9, -8.9, -8.8, \ldots, 9\}$ and dc_1 = dc_2 = dc_3= $\{-10, -9, -8, \ldots, 10\}$ It begins generating solutions, using the step range around the initial solution and calculating their RMS error. As soon as it generates a better solution, it stops, shifts the focus on that solution, resets the step range to the initial value, and continues the search in the neighborhood of the new best solution. If after four hundred thousand generated solutions no better solution is found, the step range gets doubled, and the search continues in the current neighborhood with a larger neighborhood. The stopping condition is the same as before.

Table 1: RMS error (best values) after $4 \cdot 10^6$ calculating operations

Lens/Alg.	SD	IF	RAN	IR
C13353	9.7572	**4.9422**	**5.3896**	9.2435
CA11265	4.154	**2.5374**	**3.722**	4.9367
CA11268	2.6058	**2.4788**	**2.4984**	4.0278
CA11483	**3.2673**	3.3951	**3.1944**	3.5698
CA11525	3.5799	**1.0365**	**1.4805**	2.8385
CA11934	**2.1729**	**1.4969**	2.6169	3.5317
CA12392	**1.639**	**1.5905**	1.9988	3.3103
CA13013	1.7555	**0.9042**	**1.2872**	1.7656
CP12632	**4.576**	**4.3207**	4.9078	6.7152
CP12633	7.1202	**2.936**	**2.7363**	3.8963
CP12634	**5.7641**	**5.6363**	6.1473	6.4242
CP12636	**3.1178**	**3.0801**	3.9602	4.3642
Median	3.4236	2.7367	2.9654	3.9621

5.4 Comparison of Local Search Heuristics

To reduce the performance influence of the initial solution we fixed it on all of the local search heuristics which began from the same initial solution that had the parameters set to $a_1 = a_2 = a_3 = 0.5$, $b_1 = b_2 = b_3 = 0$, and $c_1 = c_2 = c_3 = 1$. As the number of steps the local search heuristics need to find a local optima can vary heavily, it is natural to run a multi start version. As the local search runs sometimes improve the solutions in later iterations and because some preliminary experiments with multi start versions of the local search algorithms did not show any obvious advantage, we do not consider the multistart version here. However, the trivial *random search algorithm* (RAN) is included in the comparison of the local search algorithms. RAN algorithm is essentially a random solution generator that has only one simple rule. The rule is the boundary definition of the search space, so that the solutions generated stay inside the search space limits, hence RAN resembles a pure guessing exercise and any meaningful algorithm has to outperform RAN.

Table 1 and Table 2 provide best found solutions for different local search algorithms. Best two solutions are written in bold.

We can observe that most of the algorithms find a good (RMS$<5\%$) solution on almost all instances on both the long and short runs. Obviously, the majority of IF results are among the best, but also SD and RAN perform very good on some instances. Therefore we further compare the algorithms using a statistical test. We test the null hypothesis H_0: *The median of differences between results of algorithms equals 0.* The test used is a non-parametric *related samples Wilcoxon signed rank test*, see [30], and the significance level is 0.05. This means that if asymptotic significance is less or equal to 0.05, the H_0 is rejected (there is a significant difference between algorithms). Note that

Table 2: RMS error (best values) after $1.2 \cdot 10^6$ calculating operations

Lens/Alg.	SD	IF	RAN	IR
C13353	9.7572	**5.0976**	**5.3896**	10.137
CA11265	4.154	**2.5389**	**3.9455**	6.4726
CA11268	2.6058	**2.4797**	**2.4984**	4.0278
CA11483	**3.2673**	**3.4077**	3.6977	4.3763
CA11525	3.5799	**1.0381**	**1.4805**	4.4415
CA11934	**2.1729**	**1.5547**	2.6169	3.5317
CA12392	**1.639**	**1.5924**	1.9988	3.3103
CA13013	1.7555	**0.9043**	**1.2872**	2.7156
CP12632	**4.576**	**4.3292**	4.9078	6.7152
CP12633	7.1202	**2.9366**	**2.7363**	4.2827
CP12634	**5.7641**	**5.638**	6.1473	6.4713
CP12636	**3.1178**	**3.0801**	3.9602	5.2287
Median	3.4236	2.7377	3.217	4.4089

Table 3: Asymptotic significances of Wilcoxon signed rank test for results of local search heuristics at $4 \cdot 10^6$ calculating operations

	SD	IF	RAN	IR
SD		**0.007**	0.388	0.136
IF			**0.008**	**0.002**
RAN				**0.002**

the same test will be repeated on the other versions of algorithms further down the text.

Tables 3 and 4 confirm that IF significantly outperforms the other algorithms on the dataset. Also we can observe that the null hypothesis could not be rejected between RAN and SD. However we can see a significant deviation in the result of the IR algorithm which is the worst of the algorithms having the median value of $M_d = 3,9621$.

We conclude that in both the long and short runs algorithm IF prevails. Therefore IF will be our choice when infusing local search in the hybrid genetic algorithm.

6 Genetic Algorithms

The search for a more advanced heuristic method resulted in a very large pool of promising alternatives such as particle swarm optimization [19], firefly algorithm [28, 10], bat algorithm [9] and of course the well known genetic algorithms to name just a few of them. In this experimental study we use a *standard genetic algorithm* (SGA) [23] and a *hybrid genetic algorithm* (HGA) [15] that in fact mimics the evolutionary behavior [12, 20, 23], but is enhanced at every generation with the use of a local search algorithm. Encouraging preliminary results with HGA are reported in [15]. We wish to note that in the conference paper [13] local search based heuristics were compared to another version of genetic algorithm. As the algorithm in [13] used nonstandard genetic operators, it has been argued that the results are not very useful, and hence we decided to do another comparison including SGA. We wish to note that the experimental results given in this paper may slightly differ to preliminary reports [13, 15] because the preliminary results were performed on various computers, and the new experiment reported here is completely rerun on the same machine. Also, parts of the code were rewritten in order to be more computer and system independent.

Table 4: Asymptotic significances of Wilcoxon signed rank test for results of local search heuristics at $1.2 \cdot 10^6$ calculating operations

	SD	IF	RAN	IR
SD		**0.008**	0.695	**0.034**
IF			**0.004**	**0.002**
RAN				**0.002**

6.1 Standard Genetic Algorithm (SGA)

In our genetic algorithms we use three genetic operators: selection, cross-breading and mutation. The selection [12] operator works as a kind of a filter where more fitter individuals in a population get to have higher weights as the less fitter. This is then transmitted to the cross-breading operator in the way that the individuals with higher weights are more likely to be chosen as parents.

The cross-breading or crossover operator [12, 20, 23] is where a population is created by generating new solutions. These are created by randomly combining and crossing parameters from two randomly chosen parent solutions from the current population. The crossing is done via cross point so that every parent pair produces a pair of children. The cross point is chosen randomly and the children are generated in the following sequence $C_1 = [P_1^{bCP}, CP, P_2^{aCP}]$ and $C_2 = [P_2^{bCP}, CP, P_1^{aCP}]$, where C_n is the child being generated, CP is the cross point parameter, P_n^{aCP} are all of the parents parameters that are after the CP and P_n^{bCP} are all of the parents parameters that are before the CP.

The last operator in every generation is the self adapting mutation[23] operator which finalizes the individuals in the new population. The mutation operates in the following manner: in the randomly chosen individual, a random number of parameters are chosen to be changed (mutated) which is done by adding a randomly chosen value for $da_1 = da_2 = da_3 = \{-0.01, -0.009, -0.008, \ldots, 0.01\}$, for $db_1 = db_2 = db_3 = \{-0.25, -0.24, -0.23, \ldots, 0.25\}$ and $dc_1 = dc_2 = dc_3 = \{-2.5, -2.4, -2.3, \ldots, 2.5\}$ to the current parameter value.

The whole algorithm then begins with the generation and calculation of the initial population (the zero population). Next it sorts the population entities from the fittest to the least fit and applies weights to them. After the sorting process the algorithm generates with the crossover operator the next generation, which is then submitted to mutation with the adaptive mutation operator. When the new generation is fully formed the algorithm begins the process from

the point of selection. It continues to do so until the last generation is finalized. In order to assure comparable running times, the number of generations to be generated is calculated as the quotient of the maximal number of iterations minus the population size and the population size $N_G = (T_{max} - N_P)/N_P$. Where N_G stands for number of generations, T_{max} for the total number of iterations and N_P for the number of individuals in each generation (population size).

6.2 Hybrid Genetic Algorithm (HGA)

To test our theory of an advanced genetic algorithm we altered the standard genetic algorithm in a way that we infused a local optimization as an operator in every generation. We call the modified algorithm *hybrid genetic algorithm* (HGA). The hybrid genetic algorithm works in the same way as the standard one but with an extra operator before the crossover. It starts with generating the initial solution and sorts the entities in the current solution from the fittest to least fit. Then instead of directly cross breading the new generation it first runs the iterative improvement with fixed neighborhood algorithm on 10 best entities of the current generation which in turn get locally optimized (enhanced) for a number of iterations. After that the HGA follows the same path as the standard genetic algorithm does. For the number of generations to be executed on HGA algorithm, the formula is a bit more complicated, because it has to include the iterations of the local search. The formula can be written as $N_G = (T_{max} - N_P)/(N_P + 10 * N_{lo})$. Where the additional parameter N_{lo} stands for number of local search iterations. The result has to be rounded, because the algorithm cannot stop in the middle of a generation evaluation. For example if you would calculate the number of generations for the HGA13 with the above formula you would get 9.972 which gets rounded to 10. This is the reason for the minor deviation ($div_{max} = 2,5\%$) of the overall T_{max} on the HGA algorithms.

7 Parameters of the Genetic Algorithms

In order to enable fair comparison among various heuristics, the same runtime was given to all competitors. As the wallclock runtime can depend heavily on particular implementation, we measure runtime in so called basic time steps. One step of local search algorithm is naturally defined as a generation (and handling) of one feasible solution. For the genetic algorithms, time needed for the basic operations is estimated in terms of local search basic steps. This is explained in detail in the first subsection. Genetic algorithms are divided into four groups depending on the time allowed for local search improvement of the members of population.

We fix the length of the local search runs and then look for most suitable parameters of the particular HGA version.

Table 5: Parameter combinations for SGA*

Algorithm	# pop.	# gen.	# LS iter.
SGA 1	1000	3999	NA
SGA 2	5000	799	NA
SGA 3	10000	399	NA
SGA 4	50000	79	NA
SGA 5	100000	39	NA

Table 6: Parameter combinations for HGA*1

Algorithm	# pop.	# gen.	# LS iter.
HGA 1 1	1000	40	10000
HGA 2 1	5000	38	10000
HGA 3 1	10000	36	10000
HGA 4 1	50000	26	10000
HGA 5 1	100000	20	10000

This gives rise to four groups of algorithms: SGA, HGA*1, HGA*2, and HGA*3. Tables 5, 6, 7, and 8 give different parameter combinations for the genetic algorithms.

Tuning of other parameters of genetic algorithms is explained in detail below.

7.1 Runtime

To be able to compare the genetic algorithm performance to the local search algorithms we locked the total amount of computation iterations (one computation iteration in our case is the evaluation of the RMS error at the given coefficient values) on the genetic algorithms to four million on the long run and 1.2 million on the short runs, as it was in the local search algorithms. We then chose different population sizes and calculated the number of generation and local search iterations needed to achieve the desired four million calculation iterations as close as possible (minor deviations can occur due to the restriction that we are always evaluating a whole generation).

7.2 Parameter Tuning

In order to perform the final experiment we first have to choose the algorithms that would be competing in the experiment. As it would be unfeasible to compare all of the possible variations of the multi start genetic algorithms, we formed four groups, as presented in the previous section. We applied the statistical test on these groups and accord-

Table 7: Parameter combinations for HGA*2

Algorithm	# pop.	# gen.	# LS iter.
HGA 1 2	1000	20	20000
HGA 2 2	5000	19	20000
HGA 3 2	10000	19	20000
HGA 4 2	50000	16	20000
HGA 5 2	100000	13	20000

Table 8: Parameter combinations for HGA*3

Algorithm	# pop.	# gen.	# LS iter.
HGA 1 3	1000	10	40000
HGA 2 3	5000	10	40000
HGA 3 3	10000	10	40000
HGA 4 3	50000	9	40000
HGA 5 3	100000	8	40000

Table 9: RMS error (best values) after $4 \cdot 10^6$ calculating operations for SGA*

Lens/Alg.	SGA1	SGA2	SGA3	SGA4	SGA5
C13353	9.3526	8.0242	9.3895	**4.8163**	**4.848**
CA11265	5.2983	5.2999	3.8483	**3.0568**	**2.8673**
CA11268	4.5748	3.3628	2.7874	**2.7845**	**2.5248**
CA11483	3.8848	3.8262	3.8775	**3.6547**	**3.5613**
CA11525	4.0658	1.6667	2.2655	**2.0129**	**1.2501**
CA11934	2.5642	3.3076	3.3842	**1.8616**	**2.1933**
CA12392	3.493	2.5458	**2.376**	**2.3139**	2.4982
CA13013	2.9739	**1.1658**	1.4496	1.2332	**1.128**
CP12632	4.3657	4.5959	5.9514	4.4332	4.4827
CP12633	4.447	3.3542	2.855	**2.5418**	**2.5485**
CP12634	5.7747	5.7038	**5.6663**	5.7493	**5.6712**
CP12636	4.2818	4.1577	3.8485	**3.5941**	**3.491**
Median	4.3238	3.5945	3.6163	2.9207	2.7079

ing to the results chose the one that would advance into the final experiment.

7.3 SGA* Test

When observing the test results presented in Table 11 and Table 12 we see that SGA1 in both the long and short runs statistically differs from the other four algorithms. It also has the worst median where $M_d = 4.3238$. We could not reject the null hypothesis in the case of SGA2 and SGA3 on the long run and on the short run between SGA2, SGA3 and SGA5. The long run shows us shared leadership between SGA4 and SGA5, but in the long run the SGA5 prevailes with the overall best median of $M_d = 2.7079$. This leads us to ultimately choose the SGA5 as the representative of the standard genetic algorithms in the final experiment.

7.4 HGA*1 Test

The statistical results show that there in no significant statistical difference between the HGA*1 algorithms (the null hypothesis could not be rejected). Because of that we have to take a look at the median values. In both runs HGA41 had the best median value which was a bit lower on the long run $M_d = 2.5840$. Hence we chose the HGA41 algorithm from this group to advance in the final experiment.

Table 10: RMS error (best values) after $1.2 \cdot 10^6$ calculating operations for SGA*

Lens/Alg.	SGA1	SGA2	SGA3	SGA4	SGA5
C13353	9.3526	8.0242	9.3895	**6.3401**	**5.7489**
CA11265	5.2983	5.2999	**3.8483**	**3.3367**	3.933
CA11268	4.5748	3.3628	**2.7874**	**2.7845**	2.8694
CA11483	3.8848	3.8262	3.8775	**3.6547**	**3.6344**
CA11525	4.0658	**1.6667**	2.2655	**2.0129**	2.0875
CA11934	**2.5642**	3.3076	3.3842	**2.4779**	2.8535
CA12392	3.493	2.5458	**2.376**	**2.3139**	2.5325
CA13013	2.9739	**1.1658**	1.4496	**1.2493**	1.4786
CP12632	**4.3657**	4.5959	5.9514	**4.4332**	4.807
CP12633	4.447	3.3542	2.855	**2.6167**	**2.5485**
CP12634	5.7747	**5.7038**	**5.6663**	5,7493	5.7481
CP12636	4.2818	4.1577	**3.8485**	**3,5941**	3.8838
Median	4.3238	3.5945	3.6163	3.0606	3.2519

Table 11: Asymptotic significances of Wilcoxon signed rank test for results of SGA* at $4 \cdot 10^6$ calculating operations

SGA*	1	2	3	4	5
1		**0.034**	0.071	**0.004**	**0.004**
2			0.937	**0.019**	**0.002**
3				**0.005**	**0.005**
4					0.272

7.5 HGA*2 Test

In the HGA*2 group the HGA12 significantly differs from the rest of the group in both runs, with the worst median of $M_d = 3.1785$. The rest of the algorithms perform pretty much the same, so there is no visible difference between them in the long run and a slight inconclusive difference in the short run. Therefore we once again chose the algorithm to be advanced to the final experiment based on the minimal median value. The HGA42 algorithm has the best overall median value of $M_d = 2.7805$ and consequently is the one which represents this group in the final experiment.

7.6 HGA*3 Test

The last group's results are similar to the previous two. On the long run the HGA13 algorithm shows no significant difference from the others. There is also no significant statis-

Table 12: Asymptotic significances of Wilcoxon signed rank test for results of SGA* at $1.2 \cdot 10^6$ calculating operations

SGA*	1	2	3	4	5
1		**0.034**	0.071	**0.004**	**0.015**
2			0.937	**0.019**	0.117
3				**0.005**	0.158
4					0.084

Table 13: RMS error (best values) after $4 \cdot 10^6$ calculating operations for HGA*1

Lens/Alg.	HGA11	HGA21	HGA31	HGA41	HGA51
C13353	5.9871	**3.8263**	3.8339	**3.5616**	5.4869
CA11265	2.9985	2.8531	3.025	**2.824**	**2.8131**
CA11268	2.5578	**2.3826**	2.5418	2.5907	**2.3417**
CA11483	**3.1709**	**3.1571**	3.2288	3.2077	3.6024
CA11525	**1.4021**	**1.3965**	1.5203	1.5096	1.4318
CA11934	3.0945	2.1895	2.5715	**1.8058**	**1.9126**
CA12392	2.3158	2.365	2.3345	**2.038**	**2.0077**
CA13013	1.3588	**1.0952**	1.2681	**0.9672**	1.1257
CP12632	**4.38**	4.3803	**4.3495**	4.3852	4.402
CP12633	3.1962	**2.5102**	2.8362	**2.5773**	2.6248
CP12634	5.7581	**5.6829**	**5.6919**	5.7342	5.7757
CP12636	**2.288**	3.0882	**2.3877**	2.5551	2.8276
Median	3.0465	2.6816	2.7038	2.5840	2.7189

Table 14: RMS error (best values) after $1.2 \cdot 10^6$ calculating operations for HGA*1

Lens/Alg.	HGA11	HGA21	HGA31	HGA41	HGA51
C13353	6.1767	**4.3002**	4.3135	**3.8082**	5.4869
CA11265	3.2252	3.3075	3.4191	**3.1401**	**2.8131**
CA11268	2.5578	**2.3826**	2.5418	2.6582	**2.3417**
CA11483	**3.2573**	3.4174	3.3694	**3.2284**	3.6024
CA11525	**1.4021**	**1.4084**	1.5203	1.5822	1.4318
CA11934	3.0945	2.4831	3.0026	**1.8058**	**1.9126**
CA12392	2.4432	2.365	2.3345	**2.038**	**2.0077**
CA13013	1.5113	1.3467	1.4484	**1.1819**	**1.1989**
CP12632	4.5072	4.5397	**4.3849**	**4.3852**	4.5023
CP12633	3.2601	**2.6138**	3.0894	**2.5944**	2.8204
CP12634	5.8198	**5.7005**	**5.6919**	5.9876	5.7757
CP12636	**2.288**	3.0882	2.9039	**2.5551**	2.8276
Median	3.1599	2.8509	3.0459	2.6262	2.8167

Table 15: Asymptotic significances of Wilcoxon signed rank test for results of HGA*1 at $4 \cdot 10^6$ calculating operations

HGA*1	1	2	3	4	5
1		0.060	0.347	0.136	0.239
2			0.117	0.347	0.814
3				0.099	0.695
4					0.117

Table 16: Asymptotic significances of Wilcoxon signed rank test for results of HGA*1 at $1.2 \cdot 10^6$ calculating operations

HGA*1	1	2	3	4	5
1		0.239	0.583	0.158	0.099
2			0.308	0.136	0.48
3				**0.041**	0.239
4					0.583

tical difference between HGA23 and HAGA53. The best two on the long run are HGA33 and HGA43, comparing the medians. The short run results confirm the picture we get from the long run. The HGA13 and HGA23 do not significantly differ, and are the worst in the group. Also there is no significant difference between HGA33, HGA43 and HGA53. As above we choose the winner based on the median values. The best median value result was obtained by the HGA43 algorithm $M_d = 2.6589$.

8 Final Experiment

Based on statistical tests on four groups of genetic algorithms we acquired four winning algorithms, with seemingly best tuned parameters inside each group. The final experiment will compare those four algorithms with the best local search algorithm IF. Table 25 show the lowest

Table 17: RMS error (best values) after $4 \cdot 10^6$ calculating operations for HGA*2

Lens/Alg.	HGA12	HGA22	HGA32	HGA42	HGA52
C13353	4.7054	4.2986	4.1253	**3.3723**	**3.4518**
CA11265	3.796	**2.9932**	**2.9978**	3.0308	3.0097
CA11268	2.5909	2.5584	**2.4421**	2.6774	**2.5356**
CA11483	3.2911	3.2074	**3.1022**	3.3761	**3.1906**
CA11525	1.6185	**1.4497**	1.5384	**1.356**	1.5003
CA11934	3.0659	2.7858	2.9067	**2.5005**	**2.2335**
CA12392	2.3725	**2.0664**	2.3302	**2.2183**	2.4562
CA13013	1.1562	1.2387	**0.9072**	**1.0672**	1.1936
CP12632	**4.4391**	4.5954	5.1023	4.4996	**4.3441**
CP12633	2.7446	**2.5122**	2.6415	**2.5266**	2.576
CP12634	5.7346	5.7481	**5.7068**	**5.7009**	5.8207
CP12636	4.2152	**3.1483**	3.2091	**2.8835**	4.1649
Median	3.1784	2.8895	2.9522	2.7804	2.7929

Table 18: RMS error (best values) after $1.2 \cdot 10^6$ calculating operations for HGA*2

Lens/Alg.	HGA12	HGA22	HGA32	HGA42	HGA52
C13353	5.3734	4.4111	4.1253	**3.3723**	**3.7201**
CA11265	4.061	**2.9932**	3.3168	**3.1738**	3.2775
CA11268	2.7644	2.5751	**2.4421**	2.7098	**2.5356**
CA11483	3.7451	**3.2304**	3.7034	3.3761	**3.2599**
CA11525	1.6185	1.5703	1.5384	**1.356**	**1.5177**
CA11934	3.3468	**2.7858**	3.0841	**3.0173**	3.1701
CA12392	2.4229	**2.1467**	2.4255	**2.2183**	2.4562
CA13013	1.1562	1.2387	**1.1321**	**1.1125**	1.1936
CP12632	**4.4624**	4.5954	5.3015	4.6303	**4.356**
CP12633	2.7446	**2.5122**	2.8733	**2.5266**	2.588
CP12634	**5.7346**	5.7481	**5.7068**	5.7586	5.8207
CP12636	4.2152	3.4343	**3.2091**	**2.8969**	4.2121
Median	3.5459	2.8895	3.1466	2.9571	3.2149

Table 19: Asymptotic significances of Wilcoxon signed rank test for results of HGA*2 at $4 \cdot 10^6$ calculating operations

HGA*2	1	2	3	4	5
1		**0.019**	**0.023**	**0.019**	**0.023**
2			0.638	0.239	0.937
3				0.182	0.754
4					0.754

RMS errors at four million calculating iterations for every competing algorithm. Table 26 show the lowest RMS errors at bout one million calculating iterations for every competing algorithm. The asymptotic significances of related samples Wilcoxon signed rank test for results are shown in Table 27 and Table 28.

We can see a significant deviation in the result of the SGA5 algorithm which is the worst of the algorithms having the median value of $M_d = 3.2519$ on the short run. In short runs SGA5 significantly differs to all other algorithms, hence is clearly the worst among the competitors. However, there is no significant difference between algorithms HGA41, HGA42, HGA43 and IF. Simply counting the number of emphasized results (best two on particular instance) favourizes IF ($8 + 8 = 16$), followed by HGA41 ($6 + 7 = 13$). The other two only have eight emphasized results: HGA42 ($3 + 5 = 8$) and HGA43 ($4 + 4 = 8$). So we conclude that on the dataset used here the best are

Table 20: Asymptotic significances of Wilcoxon signed rank test for results of HGA*2 at $1.2 \cdot 10^6$ calculating operations

HGA*2	1	2	3	4	5
1		**0.015**	0.084	**0.008**	**0.019**
2			0.182	0.875	0.347
3				**0.019**	0.814
4					0.209

Table 21: RMS error (best values) after $4 \cdot 10^6$ calculating operations for HGA*3

Lens/Alg.	HGA13	HGA23	HGA33	HGA43	HGA53
C13353	4.7929	7.4605	4.1179	**3.673**	**3.3926**
CA11265	3.5219	**2.8982**	**3.2986**	3.3426	3.4874
CA11268	2.4712	2.6368	**2.6012**	2.4733	**2.5077**
CA11483	**3.1692**	3.9305	3.3611	3.5551	**3.3291**
CA11525	1.924	1.7202	**1.3482**	**1.6992**	1.8313
CA11934	**2.7798**	3.1422	2.9557	**2.1928**	3.1123
CA12392	2.4091	2.379	2.3175	**1.9166**	**2.3143**
CA13013	1.4897	1.2844	**1.1117**	**1.031**	1.7124
CP12632	4.5923	4.8078	**4.4902**	**4.424**	4.5191
CP12633	2.7836	2.6338	**2.5233**	**2.4582**	2.7832
CP12634	5.806	**5.7097**	**5.8002**	5.8213	5.9054
CP12636	**2.617**	4.2187	3.3837	**2.8446**	3.2144
Median	2.7817	3.0202	3.1271	2.6589	3.1633

Table 22: RMS error (best values) after $1.2 \cdot 10^6$ calculating operations for HGA*3

Lens/Alg.	HGA13	HGA23	HGA33	HGA43	HGA53
C13353	5.6511	7.8509	5.0076	**4.6041**	**3.3926**
CA11265	3.5219	**3.4391**	**3.3527**	3.5108	3.4874
CA11268	2.7614	2.761	2.6012	**2.4733**	**2.5077**
CA11483	3.48	3.9305	**3.4179**	3.8551	**3.3291**
CA11525	2.1053	1.773	**1.6147**	**1.6992**	1.8313
CA11934	3.0145	3.4084	**2.9764**	**2.4433**	3.1123
CA12392	2.5199	2.4228	2.6216	**1.9166**	**2.3143**
CA13013	1.7247	1.5402	**1.1117**	**1.031**	1.7124
CP12632	4.7738	4.8078	**4.4902**	4.7755	**4.5191**
CP12633	3.5175	2.6338	**2.5233**	**2.4582**	2.7832
CP12634	5.9669	**5.7097**	**5.8002**	5.8213	5.9054
CP12636	**2.6872**	4.2187	3.4892	3.2777	**3.2144**
Median	3.2472	3.4237	3.1646	2.8755	3.1634

the algorithms HGA41 and IF which share the leadership. Finally, as the HGA41 has a slightly lower median value $M_d = 2,6263$ than the IF, the overall winner of the experiment is the HGA41 algorithm. (Note however that the last conclusion is not confirmed with statistical test.) Convergence of the best two algorithms on an instance is shown on Figure 2.

9 Conclusions

An experimental comparison of several heuristics on an engineering problem has been carried out. Among several local search heuristics, a version of iterative improvement on a suitably defined neighborhood was chosen, based on statistical test. A standard genetic algorithm, and three versions of hybrid genetic algorithms, in which members of population were improved by short runs of local search were considered. Parameters of the algorithms were tuned by running the algorithms on the dataset with several versions of parameters and the best combination of parame-

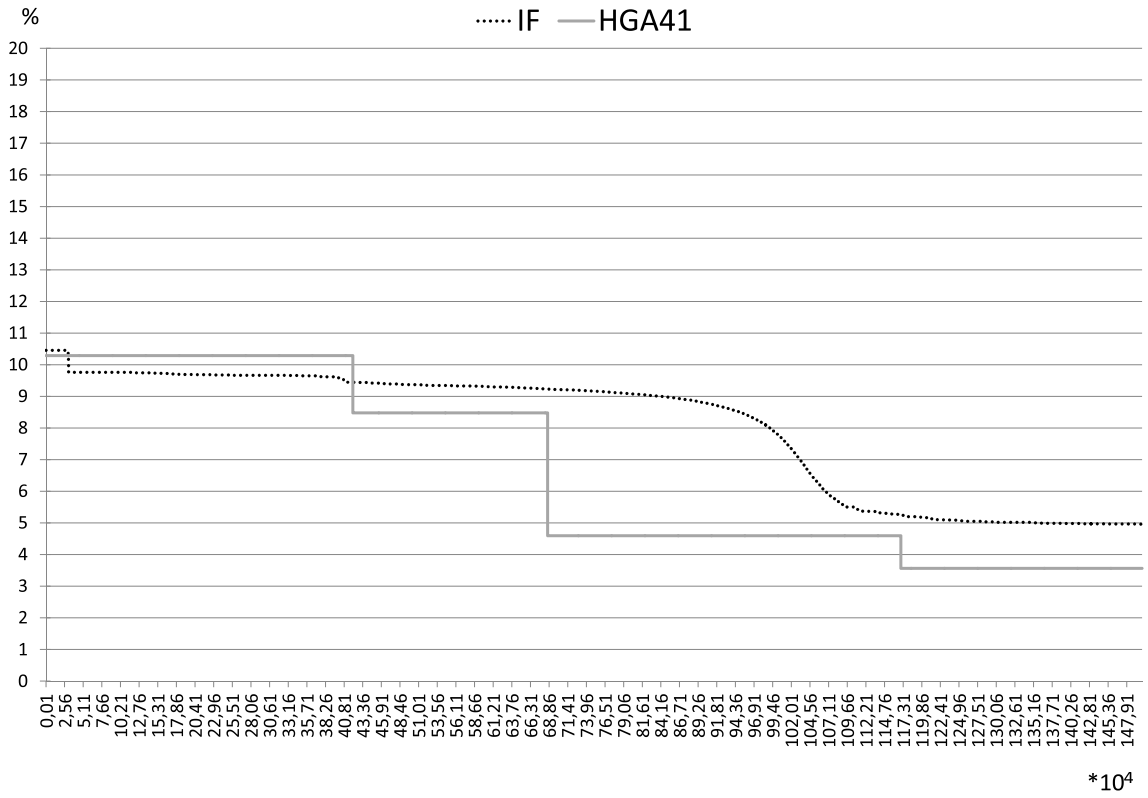

Figure 2: The convergence curves of the winning algorithms approximating the C13353 lens.

Table 23: Asymptotic significances of Wilcoxon signed rank test for results of HGA*3 at $4 \cdot 10^6$ calculating operations

HGA*3	1	2	3	4	5
1		0.308	0.347	0.084	0.48
2			**0.034**	**0.023**	0.53
3				0.099	0.347
4					**0.041**

Table 24: Asymptotic significances of Wilcoxon signed rank test for results of HGA*3 at $1.2 \cdot 10^6$ calculating operations

HGA*3	1	2	3	4	5
1		0.814	**0.05**	**0.05**	**0.05**
2			**0.015**	**0.012**	0.099
3				0.433	0.814
4					0.53

ters was selected. It may be interesting to observe that in hybrid genetic algorithms, versions with the shortest local searches were selected in all cases, meaning there is substantial number of generations possible within the runtime limit. Similarly, the standard genetic algorithm performed best when many generations were allowed and consequently, the population was smaller.

Table 25: RMS error (best values) of the final experiment after $4 \cdot 10^6$ calculating operations

Lens/Alg.	SGA5	HGA41	HGA42	HGA43	IF
C13353	4.848	**3.5616**	**3.3723**	3.673	4.9422
CA11265	2.8673	**2.824**	3.0308	3.3426	**2.5374**
CA11268	2.5248	2.5907	2.6774	**2.4733**	**2.4788**
CA11483	3.5613	**3.2077**	**3.3761**	3.5551	3.3951
CA11525	**1.2501**	1.5096	1.356	1.6992	**1.0365**
CA11934	2.1933	**1.8058**	2.5005	2.1928	**1.4969**
CA12392	2.4982	2.038	2.2183	**1.9166**	**1.5905**
CA13013	1.128	**0.9672**	1.0672	1.031	**0.9042**
CP12632	4.4827	**4.3852**	4.4996	4.424	**4.3207**
CP12633	2.5485	2.5773	**2.5266**	**2.4582**	2.936
CP12634	**5.6712**	5.7342	5.7009	5.8213	**5.6363**
CP12636	3.491	**2.5551**	2.8835	**2.8446**	3.08
Median	2.7078	2.5840	2.7804	2.6589	2.7367

In the final experiment, the best local search and the versions of genetic algorithms selected after tuning the main parameters were compared. Interesting enough, all three versions of the hybrid genetic algorithm performed better than the standard genetic algorithm, and that conclusion is supported by statistical tests. On the other hand, there is no statistically significant differences among the versions of the hybrid algorithm and the local search IF. Looking at the results closer, we conclude that a version of hybrid genetic

Table 26: RMS error (best values) of the final experiment after $1.2 \cdot 10^6$ calculating operations

Lens/Alg.	SGA5	HGA41	HGA42	HGA43	IF
C13353	5.7489	**3.8082**	**3.3723**	4.6041	5.0976
CA11265	3.933	**3.1401**	3.1738	3.5108	**2.5389**
CA11268	2.8694	2.6582	2.7098	**2.4733**	**2.4797**
CA11483	3.6344	**3.2284**	**3.3761**	3.8551	3.4077
CA11525	2.0875	1.5822	**1.356**	1.6992	**1.0381**
CA11934	2.8535	**1.8058**	3.0173	2.4433	**1.5547**
CA12392	2.5325	2.038	2.2183	**1.9166**	**1.5924**
CA13013	1.4786	1.1819	1.1125	**1.031**	**0.9043**
CP12632	4.807	**4.3852**	4.6303	4.7755	**4.3292**
CP12633	2.5485	2.5944	**2.5266**	**2.4582**	2.9366
CP12634	**5.7481**	5.9876	5.7586	5.8213	**5.638**
CP12636	3.8838	**2.5551**	**2.8969**	3.2777	3.0801
Median	3.2519	2.6263	2.9571	2.8755	2.7377

Table 27: Asymptotic significances of Wilcoxon signed rank test for results of the final experiment at $4 \cdot 10^6$ calculating operations

	SGA5	HGA41	HGA42	HGA43	IF
SGA5		0.060	0.583	0.239	**0.034**
HGA41			0.099	0.099	0.814
HGA42				0.814	0.308
HGA43					0.347

algorithm performs slightly better.

While the comparison here is based on extensive experiments and the conclusions are supported by statistical tests, there are obvious reasons that relativize the conclusions. Namely, the experiment was run on a realistic dataset that is relevant for the engineering application, and confirmed the hypothesis that these type of problems can successfully be solved by the heuristics used. On the other hand, the dataset used is relatively small, and hence the observations can be generalized only conditionally. Further work on other datasets and related optimization problems is planned.

Nevertheless, we believe that inclusion of a carefully chosen local search into a genetic algorithm is a good idea, and the present experimental study proves that.

Table 28: Asymptotic significances of Wilcoxon signed rank test for results of the final experiment at $1.2 \cdot 10^6$ calculating operations

	SGA5	HGA41	HGA42	HGA43	IF
SGA5		**0.006**	**0.008**	**0.01**	**0.005**
HGA41			0.638	0.136	0.48
HGA42				0.347	0.239
HGA43					0.099

References

[1] E. H. L. Aarts, J. K. Lenstra (1997) *Local Search Algorithms*, John Wiley & Sons.

[2] I. Ashdown (2001) Thinking Photometrically Part II., *Proceedings of the Pre-Conference Workshop (LIGHTFAIR)*.

[3] T. Bäck (1996) *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*, Oxford University Press.

[4] M. Coffin, M. J. Saltzman (2000) Statistical Analysis of Computational Tests of Algorithms and Heuristics, *INFORMS Journal of Computing*, vol. 12, pp. 24–44.

[5] Cree, Inc. *XLamp XP-E*, `www.cree.com/led-components-and-modules/products/xlamp/discrete-directional/xlamp-xpe`

[6] M. Dorigo, M. Birattari, T. Stützle (2006) Ant colony optimization, *IEEE Computational Intelligence Magazine*, vol. 1, no. 4, pp. 28–39.

[7] G. Dueck, T. Scheuer (1990) Threshold Accepting: A General Purpose Optimization Algorithm Appearing Superior to Simulated Annealing, *Journal of Computational Physics*, vol. 90, pp. 161–175.

[8] P. Festa, M. G. C. Resende (2002) GRASP: An annotated bibliography, *Essays and Surveys on Metaheuristics*, C. C. Ribeiro, P. Hansen (eds.), Kluwer Academic Publishers, pp. 325—367.

[9] I. Fister Jr., S. Fong, J. Brest, I. Fister (2014) A Novel Hybrid Self-Adaptive Bat Algorithm, *The Scientific World Journal*, vol. 2014, Article ID 709738.

[10] I. Fister, M. Perc, S. M. Kamal, I. Fister (2015) A review of chaos-based firefly algorithms: Perspectives and research challenges, *Applied Mathematics and Computation*, vol. 252, pp. 155—165.

[11] F. Glover, M. Laguna (1997) Tabu Search In M. Panos, *Handbook of Combinatorial Optimization*, New York, Springer US, pp. 2093-2229.

[12] R. L. Haupt, S. E. Haupt (2004) *Practical Genetic Algorithms*, 2nd Ed., John Wiley & Sons.

[13] D. Kaljun, J. Žerovnik (2014) Local Search Optimization of a Spatial Light Distribution Model, *Proceedings of the Student Workshop on Bioinspired Optimization Methods and their Applications (BIOMA)*, pp. 81–91.

[14] D. Kaljun, J. Žerovnik (2014) Function fitting the symmetric radiation pattern of a LED with attached secondary optic, *Optics Express*, vol. 22, pp. 29587–29593.

[15] D. Kaljun, J. Žerovnik (2014) On local search based heuristics for optimization problems, *Croatian Operational Research Review*, vol. 5, no. 2, pp. 317–327.

[16] S. Kennedy (2005) Escaping the bulb culture: the future of leds in architectual illumination. *LEDs magazine*, vol. 1, pp. 13–15.

[17] P. J. Laarhoven, E. H. Aarts (1987) Simulated annealing: theory and applications, *Mathematics and Its Applications*, M. Hazewinkel (ed.), Springer, pp. 7–15.

[18] Ledil Oy., `www.ledil.com/`.

[19] L. Lobachinsky, A. Bahabad (2014) Using Particle Swarm Optimization to Design Broadband Optical Nano-antennas for Nonlinear Optics, *Frontiers in Optics*, Optical Society of America, paper FTh4E.3.

[20] M. Mitchell (1999) *An Introduction to Genetic Algorithms*, 5th Ed., The MIT Press.

[21] N. Mladenović, P. Hansen, J. Brimberg (2013) Sequential clustering with radius and split criteria, *Central European Journal of Operations Research*, vol. 21, suppl. 1, pp. 95–115.

[22] I. Moreno, C.-C. Sun (2008) Modeling the radiation pattern of leds, *Optics Express*, vol. 16, pp. 1808–1819.

[23] D. Simon (2013) *Evolutionary Optimization Algorithms*, John Wiley & Sons.

[24] C.-C. Sun, T.-X. Lee, S.-H. Ma, Y.-L. Lee, S.-M. Huang (2006) Precise optical modeling for led lighting verified by cross correlation in the midfield region, *Optics Letters*, vol. 31, pp. 2193–2195.

[25] E.-G. Talbi (2009) *Metaheuristics: From Design to Implementation*, John Wiley & Sons.

[26] The Millennium Prize Problems are seven problems in mathematics that were stated by the Clay Mathematics Institute in 2000, `www.claymath.org/millenium-problems/p-vs-np-problem`.

[27] The Subcommittee on Photometry of the IESNA Computer Committee (2002) Iesna standard file format for the electronic transfer of photometric data and related information, Technical Report ANSI IESNA LM-63-02, Illuminating Engineering Society of North America.

[28] M. Tuba, N. Bacanin (2014) Improved seeker optimization algorithm hybridized with firefly algorithm for constrained optimization problems, *Neurocomputing*, vol. 143, pp. 197–207.

[29] A. Vesel, J. Žerovnik (2000) How well can ants colour graphs?, *CIT. Journal of Computing and Information Technology*, vol. 8, pp. 131–136.

[30] F. Wilcoxon (1945) Individual comparisons by ranking methods, *Biometrics*, vol. 1, pp. 80–83.