Croatian Operational Research Review
CRORR **5**(2014), 317–327

On local search based heuristics for optimization problems

David Kaljun[1] **and Janez Žerovnik**[2,*]

[1] *Faculty of Mechanical Engineering, University of Ljubljana*
Aškerčeva 6, 1000 Ljubljana, Slovenia
E-mail: ⟨david.kaljun@fs.uni-lj.si⟩

[2] *Faculty of Mechanical Engineering, University of Ljubljana*
Aškerčeva 6, 1000 Ljubljana, Slovenia
and
Institute of Mathematics, Physics and Mechanics
Jadranska 19, 1000 Ljubljana, Slovenia
E-mail: ⟨janez.zerovnik@fs.uni-lj.si⟩

Abstract. When comparing various metaheuristics, even asking a fair and formally consistent question is often difficult. Having this in mind, we provide some further evidence that simple local search heuristics may be at least a very competitive choice. On a dataset from an industrial application, i.e., construction of an optical system, we compare local search and genetic algorithms. In our case, the best performance is obtained by a combination of both heuristics.

Key words: optimization, heuristics, local search

Received: September 23, 2014; accepted: December 11, 2014; available online: December 30, 2014

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. From a practical point of view, knowing that the problem is computationally intractable implies that we may use heuristic approaches and that we should also aim to find nearly optimal solutions for which sometimes but not always approximation bounds can be given. Our experience (and many studies in the literature) show 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 the properties of the most likely instances. On the other hand, it is useful to work within a framework of some (one or more) metaheuristcs [24] which can be seen as general strategies to attack an optimization problem.

Perhaps the most natural and conceptually simple metaheuristics is local search. In the search space of feasible solutions the solutions with extremal values of the

*Corresponding author.

goal functions are to be found. In order to speak about local search, a topology is introduced, usually via a 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 local search, iterative improvement and best neighbor (or steepest descent). As the names indicate, starting from the initial feasible solution, iterative improvement generates a random neighbor, and moves to the new solution based on the difference in the goal function. The procedure stops when there is no improvement for a sufficiently long time. On the other hand, best neighbor heuristics (also called steepest descent) considers all neighbors and moves to the new solution with best value of the goal function. If there is no better neighbour, the current solution is clearly a local optimum. Note that given a particular optimization problem, different neighborhood structures can be defined giving rise to different local search heuristics. In fact, many metaheuristics can be seen as variations or improvement of local search [1], popular examples include simulated annealing, tabu search [9], iterated local search, variable neighborhood search [18], and GRASP (Greedy Randomized Adaptive Search Procedure) [6]. The other type of search strategy has a learning component added to the search, aiming to improve the obvious drawback of local search, a complete lack of memory. An exception is the tabu search that successfully introduces a short time memory. Metaheuristics motivated by the idea of learning from past searches include ant colony optimization [5], evolutionary computation [22], and genetic algorithms [17]. It is, however, a good question in each particular case whether learning does indeed mean an improvement [25]; namely, a successful heuristic search must have both enough intensification and diversification. The second important issue that may have an essential impact on the success of the multistart local search based optimization is the selection of the initial solution. Sometimes, for example in a real world application, we have a known practical solution that we want to improve. On the other hand, quite often it is possible to generate many initial solutions easily. In such cases, a construction that is both greedy and to some extent random, may be the winning idea. Note that the quality of the initial solution is often not essential. Usually, more important is to have a pool of reasonably good starting solutions that are at the same time randomly generated thus assisting the multistart algorithm diversification.

In this paper, it is shown by an experimental example that local search, the most basic metaheuristics, is a very competitive choice. In particular, several local search based heuristics are compared with a standard genetic algorithm. We also design a hybrid genetic algorithm where the survived population is improved by a local optimization.

The rest of the paper is organized as follows. In the next section, we introduce the practical problem that we consider. In Section 3, we give some details of the algorithms we compare. In Section 4, the results of computational experiments are outlined and in Section 5 concluding remarks are given.

2. The problem

The problem at hand is motivated by a research project in which we are trying to provide a method for goal driven optimization of the luminaire photometry. The main task is to define the combination and position of secondary optical elements on a LED array in a way that satisfies user demands on the arrays end photometry. One of the prerequisites for an efficient optimization method is that the elements used in the optimization process are described efficiently. In our case, it means that the input data should be described in a form which utilizes a small number of parameters. The standard photometric data is composed of measured candela values in all spatial directions (photometric distribution). The data is then presented as a set of vectors written in spherical coordinates in the following form [horizontal angle, polar angle, candela value]. The CIE standard requires the measured data for general purpose use to have at least 3,312 (72 horizontal angles and 46 reading on each polar angle) measured points for a full asymmetric distribution [26]. A more compact form of the data that may be manipulated easier and faster is sought. For this purpose, we utilize a mathematical model [19] that was capable to accurately describe the spatial photometric distribution of a LED element without secondary optics, with the sum of three cosine functions. We adopted the model and made slight technical changes to ease its use in our heuristics. We added the I_{max} value to the model to fix the function parameters in the same interval. Consequently, there will be three types of parameters, $a_\star \in [0,1]$, $b_\star \in [-90,90]$, and $c_\star \in [0,100]$. Note that the parameters in the model are continuous, so a feasible value can be any real number from the corresponding interval. However, in the optimization process we will use a discretization of the interval. The adopted model is given by

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

where $\mathbf{a} \in [0,1]^K$, $\mathbf{b} \in [-90,90]^K$, and $\mathbf{c} \in [0,100]^K$. Based on given data $(\Theta_i, I_m(\Theta_i))$, $I = 1,2,\ldots N$, $(N >> 3K)$ optimal parameters minimizing the function $F\left(\mathbf{a},\mathbf{b},\mathbf{c}\right) = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left[I_m(\Theta_i) - I(\Theta_i,\mathbf{a},\mathbf{b},\mathbf{c})\right]^2}$ should be determined. In order to simplify the problem, it is sufficient to consider the standard least squares problem $G\left(\mathbf{a},\mathbf{b},\mathbf{c}\right) = \sum_{i=1}^{N}\left[I_m(\Theta_i) - I(\Theta_i,\mathbf{a},\mathbf{b},\mathbf{c})\right]^2$. The problem is one of the so-called "separable nonlinear least squares problems" studied already in the 1970s [10, 15].

There are two differences as opposed to the article [19]. First is that we describe the spatial distribution of a LED and secondary optical element combo, which means that we use the model for a slightly different phenomenon. The second difference is the introduction of global I_{max}, thus changing the ranges of parameters. This only simplifies implementation while it is clear that the two models are equivalent.

From the preliminary experiment we deducted that a minimum sum of three cosine functions ($K = 3$) will be appropriate to describe the distribution accurately. This is in accordance with observations of Moreno and Sun, but it should be noted that this is only valid for the examples which are relatively simple, c.f. rotation symmetric. The method of fitting the function parameters to the given data will be

done via minimizing the standard RMS error

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

The sum runs over all N measured (or, desired) values $I_m(\Theta_i)$. Recall that this is equivalent to minimizing the sum of squares $\sum_{i=1}^{N}\left[I_m(\Theta_i) - I(\Theta_i,\mathbf{a},\mathbf{b},\mathbf{c})\right]^2$, i.e., finding parameter values $\mathbf{a}^*$, $\mathbf{b}^*$, and $\mathbf{c}^*$ for which the sum is minimal. In other words, we are looking for an unknown function $I(\Theta) = I(\Theta,\mathbf{a}^*,\mathbf{b}^*,\mathbf{c}^*)$ defined on interval $[0,90]$ such that it fits the given values in points $\Theta_i = i * 90/(N-1)$, $i = 0,1,2,\ldots,N-1$ as good as possible, i.e., it minimizes the sum of squares of errors in the observed points. (In our dataset, $N = 46$.)

In practical application, the RMS value for a sufficiently accurate fit must be less than 5%. This is because the 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 less than 1% or 2% RMS error.

To sum up, given the standard photometric dataset (3,312 triples), our problem is to find a vector of parameters $t = (a_1,a_2,a_3,b_1,b_2,b_3,c_1,c_2,c_3)$ with the minimal RMS value. The space of feasible solutions is thus a Cartesian product of 9 intervals. Below we will work with a finite subset of parameter values, in particular $a_\star \in [0, 0.001, 0.002, \ldots, 1]$, $b_\star \in [-90, -89.9, -89.8, \ldots, 90]$, and $c_\star \in [0,1,2,\ldots,100]$. Hence, the discrete search space here consists of $N_t = 1000^3 * 1800^3 * 100^3 \sim 5,83 * 10^{24}$ tuples $t = (a_1,a_2,a_3,b_1,b_2,b_3,c_1,c_2,c_3)$.

3. Local search and genetic algorithms

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. Of course, there may be other neighborhoods that would be worth considering. The reason we keep the selected neighborhoods and do not try to look for other possibilities is simply the fact that they have already given us results of sufficient quality.

In the experiment and in the study, we address the optimization problem as a discrete optimization problem. Natural questions that may be asked here are why we use heuristics and why discrete optimization heuristics on a continuous optimization problem. First, the application of an approximation method is justified because there is no analytical solution to the best approximation of this type of functions. More precisely, we are not aware of any analytical solution, and since the functions used in the model are relatively complicated, we are not surprised that we could not find any analytical solution of the problem in the literature. Second, in order to apply continuous optimization methods such as the Newton method, we would usually need a good approximation in order to assure convergence. More precisely, the

convergence theorems usually assume certain conditions the initial condition must satisfy. Therefore, a method for finding a good starting solution before running a fine approximation based on continuous optimization methods is needed. However, in view of the at least 2% noise in the data, these starting solutions in our case may in many cases already be of sufficient quality! Nevertheless, a fine approximation based on continuous optimization methods could be used as postprocessing. 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. We plan to do it in future because it may be a useful method in applications where more demanding approximation rates are needed.

We have started our experiments with two basic local search algorithms, steepest descent (SD) and iterative improvement (IF), whereby in both cases the neighborhoods were defined in the same way. We call this neighborhood a fixed step size neighborhood. The third local search algorithm (IR) is a variation of iterative improvement, where we introduce a random step size; roughly speaking, given a step size and direction as before, we randomly make a step in the direction that is at most as long as in the fixed size neighborhood search. More precisely, the local optimization algorithms are defined as follows.

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 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 is a neighbor with a better RMS value, the search moves to the neighbor with the minimal RMS value (if there are more minimal neighbors, any of them is chosen with the same probability). If none of 512 is better than the current solution, a new set of neighboring solutions are generated, this time with a double step. It repeats for ten steps and if there is still no better solution, it breaks the search, multiplies the step value by 0.9, so the step is finer, and begins the search from the start in the neighborhood of the current solution. The algorithm stops when the number of generated solutions reaches T_{max}.

The **iterative improvement with fixed neighborhood (IF)** algorithm initializes the same neighborhood as SD. Instead of considering all 512 neighbors at once, the algorithm generates a neighbor randomly, and moves to the neighbor if its RMS value is better than the current RMS value. If no better neighbor is found after 1,000 trials, it is assumed that no better neighbor exists. As above, the algorithm changes to a new neighborhood, this time with a double step. It goes for ten steps and if there is still no better solution, it breaks the search, multiplies the step value by 0.9, so the step is finer and begins the search from the start in the neighborhood of the current solution. The algorithm stops when the number of generated solutions reaches T_{max}.

The **iterative improvement with a variable neighborhood (IR)** algorithm begins as the previous two algorithms. It initializes the same initial function parameter values. Next, it initializes 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 400,000 generated solutions no better solution is found, the step range gets doubled, and the search continues.The stopping condition is the same as before.

Naturally, whenever local search is used, the multi start version is worth considering. As preliminary testing of the multi start version was not competitive with single longer runs, we decided to use a more advanced heuristics that would, on the one hand, take advantage of the seemingly successful local search and possibly accumulate information obtained by independent local searches. Our choice was to use a **standard genetic algorithm (SGA)** that in fact mimics the evolutionary behavior [11, 17, 22]. Three genetic operators are used, i.e., the selection [11], where the more fitter in a population get to be chosen as parents more likely than the less fitter, crossover [11, 17, 22] or breading, where a new solution is created by randomly combining and crossing parameters from two randomly chosen solutions (parents) (crossing is done via a cross point so that every parent pair produces a pair of children) and the last operator in every generation is self-adapting mutation [22] operator which operates in the following manner: in the randomly chosen individual, a random number of parameters is 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 standard genetic algorithm begins with the generation and calculation of the initial population (the zero population). Then it sorts the population entities from the fittest to the least fit. After the sorting process, with the crossover operator the algorithm generates the next generation, which is then submitted to mutation with the adaptive mutation operator. For a second run of evolutionary algorithms we decided to alter the standard genetic algorithm in a way that we infused a local optimization in every generation. We called the modified algorithm a **Hybrid genetic algorithm (HGA)**. As implied above, the hybrid genetic algorithm works in the same way as the standard one but with an extra operator before the crossover operator. It starts with generating the initial solution and sorts the entities in the current solution from the fittest to the least fit. Then instead of directly cross breading the new generation, it first runs the iterative improvement with the fixed neighborhood algorithm on 10 best entities of the current generation which in turn get locally optimized (enhanced). After that, there follows the same path that the standard genetic algorithm does. As for other algorithms the stopping condition is based on the number of generated solutions.

The parameters used in the experimental study presented below were as follows. SGA: population size 20,000, number of generations 199. HGA: population 20,000, 33 generations, ten best individuals in each generation were improved by a 10,000 iterations long local search. These parameter values were chosen based on general recommendations for genetic algorithms, and were not tuned for this particuar application.

4. Experimental results

Before we dive into the result, we have to note that genetic algorithms have changed in contrast to the work presented on the workshop BIOMA 2014 [14], where we have implemented non-standard operators for genetic algorithms, which in turn made it harder to compare our algorithm to the existing ones. A complete batch of experiments was repeated which also contributed to slightly different results as presented in [14], because the new experiment was run on different computers with different architecture. Besides [14], we are not aware of any other experimental or computational study of the problem; in particular, [19] only proposes the model and does not address the question how the fitting can be obtained efficiently.

To avoid trivialities, we also compare results obtained by a simple generation of **random solutions (RAN)**.

Below we discuss the results of a comparative experiment in which all the algorithms were run with $T_{max} = 4$ million. Here T_{max} stands for the number of basic steps (atomic operations, i.e., the number of feasible solutions generated. For the purpose of algorithm evaluation, we have chosen a set of real available lenses to be approximated. The set was chosen from the online catalogue of one of the biggest and most present manufacturer in the world Ledil Oy Finland [9]. Choosing from the broad spectrum of lenses in the catalogue was based on the decision that the used LED is Cree XP-E [4], and the demand that the lenses have a symmetric spatial light distribution. We have preserved the lens product codes from the catalogue, so the reader can find the lens by searching the catalogue for the code from the first column in Table 1.

Lens/Alg.	SD	IF	RAN	IR	HGA	SGA
C13353	9.757	**4.942**	9.243	5.389	**5.076**	8.531
CA11265	2.775	**2.372**	4.936	4.798	**2.729**	4.259
CA11268	**2.227**	**2.229**	4.100	2.471	2.578	2.742
CA11483	**3.100**	**3.066**	4.130	3.387	3.141	3.867
CA11525	3.150	**1.108**	3.217	1.907	**1.087**	2.175
CA11934	3.940	**2.514**	4.196	3.543	**2.909**	3.346
CA12392	**1,636**	**1,641**	3,424	2,445	2,277	2,395
CA13013	1.202	**0.695**	2.136	2.241	**0.916**	0.932
CP12632	5.537	5.493	**4.918**	4.974	**4.362**	4.481
CP12633	2.431	**2.415**	4.063	3.708	**2.347**	2.496
CP12636	**2.348**	**2.107**	4.571	4.217	2.479	4.299
FP13030	**2.267**	**2.257**	3.762	3.659	2.414	2.749

Table 1: *RMS error after 4 million calculating operations, best of 10 runs.*

Table 1 gives the overall best solutions after the long runs of all algorithms on all twelve instances from the dataset. Recall that the results are acceptable if they have RMS values lower than 5% and that the approximation better than 1% does not bring any additional value because of the noise in data. The best two results for each instance are given in bold. First, observe that all algorithms in most of the cases give acceptable results, i.e., lower than 5 which is the same as 5% recalling the meaning of the normalizing parameter I_{max}. If we take a closer look at the values, we can see that the iterative improvement with fixed size IF is the winner when counting the number of best solutions, achieving the best solution in six out of twelve instances.

The second best is the hybrid genetic algorithm with four best solutions, followed by the steepest descent with two. Second, comparing the three local search algorithms and the genetic algorithm in terms of the quality of their best solutions on particular instances, we see that all best solutions are within 1%. We can conclude that all four are fairly comparable in terms of the expected quality of the solution. On the other hand, the blind random search on average does not produce as good results as the other four; however, it may luckily guess good solutions, in one case even the best solution was obtained (instance CP12632).

Lens/Alg.	SD	IF	RAN	IR	HGA	SGA
C13353	9.757	9.167	10.950	**5.389**	**8.477**	8.966
CA11265	**3.477**	**2.700**	7.282	5.073	4.183	5.883
CA11268	**2.376**	2.620	5.893	**2.471**	2.932	2.996
CA11483	4.181	**3.400**	4.130	3.784	**3.641**	4.027
CA11525	3.813	**3.395**	4.811	3.789	**1.601**	2.175
CA11934	4.032	**1.662**	4.988	**3.543**	3.789	4.473
CA12392	**1.814**	**1.661**	3.597	2.717	2.577	3.867
CA13013	2.804	3.115	**2.136**	2.241	**1.331**	3.558
CP12632	9.501	9.839	8.474	**5.054**	**4.703**	5.474
CP12633	**2.465**	4.511	4.757	4.296	**2.613**	3.918
CP12636	5.000	6.297	5.506	**4.217**	**3.803**	4.590
FP13030	**2.800**	5.679	6.611	3.659	**3.233**	5.363

Table 2: *RMS error after 750 thousand calculating operations, best of 10 runs.*

As we have so many results of acceptable quality, a natural question is whether the time limit chosen above could be shortened. The long runs in our implementations took 30 minutes for every run on an Intel Core I3-4130 @ 3.5 Ghz, programmed in C++ (the code was not optimized). Therefore it is interesting to compare shorter runs, see Table 2.

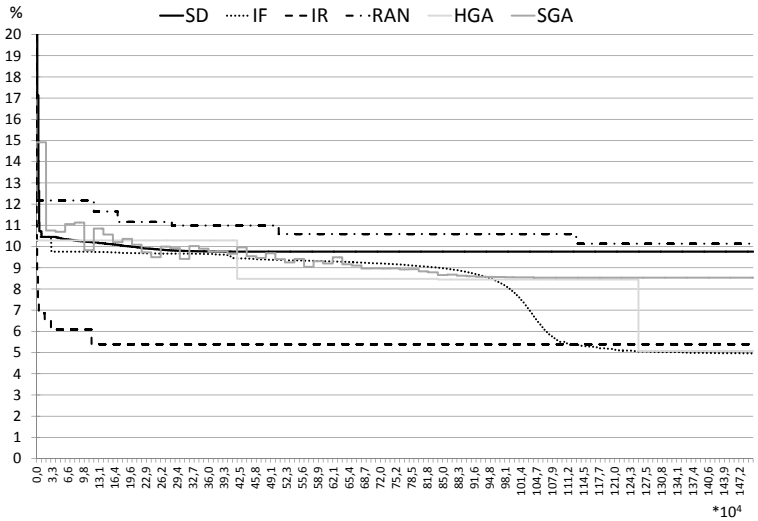

Figure 1: *Convergence curve of the fitting algorithms on the C13353.*

The shorter runs again show that most algorithms achieve the 5% error bound already in short runs. The hybrid genetic algorithm is the winner in eight out of twelve cases looking at the best obtained solution. We also observe that in short runs, the two algorithms based on fixed size neighborhood outperform the random size neighborhood iterative improvement. As expected, blind random search is not competitive on average; however curious it might be, but it is the winner on one instance. Finally, comparing the speed of convergence, we observe that all algorithms have a very steep convergence curve. A typical example is given in Figure 1.

5. Conclusions

In the previous study [14], it was observed that practicaly good solutions can be computed both by genetic algorithms and various local search based algorithms. Here we focus on an experimental comparison of local search and a standard genetic algorithm. The results can be seen as a further evidence that when considering metaheuristics, a simple local search heuristics may be at least a very competitive choice. In particular, the standard genetic algorithm in this case was clearly outperformed by a version of local search, and was comparable to the two other variants of local search. An interesting observation is that the best performance on short runs is obtained by a combination of both heuristics. In other words, a genetic algorithm was substantially better when each population was improved using local search runs.

There are many avenues of future research that are worth considering. The experimental results presented here were obtained applying the standard heuristics with ad hoc parameters. More careful tuning of parameters of all competitors may change the results dramatically. We plan to investigate combinations of parameters for genetic and hybrid algorithms in more detail in future work.

On the other hand, there is no reason to be limited by genetic algorithms and local search only. In particular, when there may exist more than one optimal solution, it may be useful to apply metaheuristics for global optimization such as the branch and bound [8, 12, 13, 21] or the DIRECT method [7, 20], as one of reviewers suggested.

Acknowledgment

This work was supported in part by ARRS, the Research Agency of Slovenia, grant P1-0285. The authors wish to thank two anonymous reviewers for their careful reading of the manuscript and constructive remarks.

References

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

[2] I. Ashdown (2001). Thinking photometrically part ii. In *LIGHTFAIR 2001 Pre-Conference Workshop*.

[3] Cree Inc. 4600 silicon drive, Durham, North Carolina 27703, USA, Technical report. `http://www.cree.com/led-components-and-modules/products/xlamp/discrete-directional/xlamp-xpe` [Accessed on 14 August 2014]

[4] S. Kennedy (2005). Escaping the bulb culture: The future of leds in architectual illumination. LEDs magazine, 1, 13–15
[5] M. Dorigo and T. Stutzle (2004). Ant Colony Optimization. Cambridge (Massachusetts): MIT Press.
[6] T.A. Feo and M.G.C. Resende (1995). Greedy randomized adaptive search procedures. Journal of Global Optimization, 6, 109–133.
[7] D. E. Finkel and C. T. Kelley (2006). Additive scaling and the DIRECT algorithm. Journal of Global Optimization, 36, 597–608.
[8] C.A. Floudas and C.E. Gounaris (2009). A review of recent advances in global optimization. Journal of Global Optimization, 45, 3–38.
[9] F. Glover and M. Laguna (1997). Tabu Search. Dordrecht: Kluwer Academic Publishers.
[10] G.H. Golub, and V. Pereyra (1973). The differentiation of pseudo-inverses and nonlinear least squares problems whose variables separate. SIAM Journal of Numerical analysis, 10, 413–432.
[11] R. L. Haupt and S. E. Haupt (2004). Practical Genetic Algorithms. Chichester: John Wiley & Sons.
[12] E. M.T. Hendrix, B.G. Toth (2010). Introduction to Nonlinear and Global Optimization. Berlin: Springer.
[13] D. R. Jones, C. D. Perttunen, and B. E. Stuckman (1993). Lipschitzian optimization without the Lipschitz constant. Journal of Optimization Theory and Applications, 79, 157–181.
[14] D. Kaljun and J. Žerovnik (2014). Local search optimization of a spatial light distribution model, in bioinspired optimization methods and their application, J. Šilc and A. Zamuda (eds.), Ljubljana: Jožef Stefan Institute, 81–92.
[15] L. Kaufman, and V. Pereyra (1978). A method for separable nonlinear least squares problems with separable nonlinear equality contraints. SIAM Journal of Numerical analysis, 15, 12–20.
[16] Ledil Oy. Salorankatu 10, fi-24240 salo, Finland, Technical report. `http://www.ledil.com/` [Accessed on 14 August 2014]
[17] M. Mitchell (1999). An Introduction to Genetic Algorithms. Cambridge (Massachusetts): MIT Press.
[18] N. Mladenović and P. Hansen (1997). Variable neighborhood search. Computers & OR, 24, 1097–1100.
[19] I. Moreno and C.-C. Sun (2008). Modeling the radiation pattern of leds. Optics Express, 16, 1808–1819.
[20] R. Paulavicius and J. Žilinskas (2014). Simplicial Global Optimization. Berlin: Springer-Verlag.
[21] J.D. Pinter (1996). Global Optimization in Action (Continuous and Lipschitz Optimization: Algorithms, Implementations and Applications). Dordrecht: Kluwer Academic Publishers.
[22] D. Simon (2013). Evolutionary Optimization Algorithms. Chichester: John Wiley & Sons.
[23] C.-C. Sun, T.-X. Lee, S.-H. Ma, Y.-L. Lee, and S.-M. Huang (2006). Precise optical modeling for led lighting verified by cross correlation in the midfield region. Optics Letters, 31, 2193–2195.
[24] E.-G. Talbi (2009). Metaheuristics: From Design to Implementation, Chichester: John Wiley & Sons.
[25] A. Vesel and J. Žerovnik (2000). How well can ants colour graphs? CIT. Journal of Computing and Information Technology, 8, 131–136.

[26] 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 ANSIDESNA LM-63-02, Illuminating Engineering Society of North America.