DE GRUYTER OPEN

Logistics & Sustainable Transport
Vol. 6, No. 1, November 2015, 1–10
doi: 10.1515/jlst-2015-0006

Heuristics for NP-hard optimization problems - simpler is better !?

Janez ŽEROVNIK[1,2]
[1] FME, University of Ljubljana , Slovenia
[2] IMFM, Ljubljana, Slovenia

***Abstract*— We provide several examples showing that local search, the most basic metaheuristics, may be a very competitive choice for solving computationally hard optimization problems. In addition, generation of starting solutions by greedy heuristics should be at least considered as one of very natural possibilities. In this critical survey, selected examples discussed include the traveling salesman, the resource-constrained project scheduling, the channel assignment, and computation of bounds for the Shannon capacity.**

***Key words*— Optimization, metaheuristics, local search, greedy construction, traveling salesman problem.**

I. INTRODUCTION

Transportation and location problems are among the most studied problems in combinatorial optimization and operational research. Even the most simply stated 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. The question is among the most challenging theoretical problems and was included into a list of seven millennium problems [31]. Practical problems are usually more complex as we have to take into account various additional constraints and goals when designing a model. Knowing the problem is computationally intractable implies that we may use heuristic approaches and that we also should aim to find nearly optimal solutions for which sometimes even approximation guaranties cannot be given. The present author shares the opinion 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) metaheuristics which can be seen as general strategies to attack an optimization problem. Hundreds of research papers were published on general and even on particular heuristics, for example evolutionary algorithms, ants algorithms, and even neural networks, to name just a few. In the last decades, thousands of research papers on theory and applications of heuristics have been published. Some of the popular metaheuristics, including some new or reinvented ones, are seemingly interesting because they are based on analogies to natural or social phenomena. The motivating story, and the theory behind, may sometimes be quite involved. Recall for example the simulated annealing algorithm, the evolutionary algorithms, and ants algorithms, as some of the popular examples from the past that are extensively used for optimization in current research. A fresh inexperienced reader might sometimes get the impression that a metaheuristics is more efficient when it is more complicated. This is of course not true. However, the fact that in optimization, the primary task is to find good, and possibly better, feasible solutions is indeed sometimes nearly forgotten. On the other hand, as the title of this paper says, perhaps the best heuristics are the simplest ones! In this paper, it is argued that local search, the most basic metaheuristics, is a very competitive choice. Furthermore, for generation of starting solutions, properly selected greedy heuristics should be at least considered as one of the possibilities. Examples include the traveling salesman, resource-constrained project scheduling, channel assignment, and bounds for the Shannon capacity. The paper is a revised and extended version of the author's presentation at the conference "The international Conference on Logistics and Sustainable Transport" that was held in Celje at the Faculty of Logistics in June 2014.

The rest of the paper is organized as follows. As the readers may not be specialists for metaheuristics, we start with some basic notions. First we introduce the problems in combinatorial optimization and in Section III, the heuristics and metaheuristics. In Section IV we give two examples of greedy constructions followed by local search improvement, and in Section V, we briefly discuss another simple heuristics – constant temperature simulated annealing that is again just a relaxed local search or more precisely, iterative improvement. In the last section, we summarize the discussion.

II. THE PROBLEMS

In applied mathematics and theoretical computer science, combinatorial optimization is a topic that consists of finding an optimal object from a finite set of objects [8]. In many such problems, exhaustive search is not feasible. It operates on the domain of those optimization problems, in which the set of feasible solutions is discrete or can be reduced to discrete, and in which the goal is to find the best solution. Combinatorial optimization is closely related to operations research, algorithm theory, and computational complexity theory. In [30] the list of applications for combinatorial optimization includes: developing the best airline network of spokes and destinations, deciding which taxis in a fleet to route to pick up fares, determining the optimal way to deliver packages, determining the right attributes of concept elements prior to concept testing, etc. Hence there is no need for more arguments to conclude that logistics is a natural source of applications for combinatorial optimization.

In order to illustrate our main idea we will mention the traveling salesman problem (TSP), which is perhaps the most studied problem in combinatorial optimization. In lecture notes for students, we sometimes relate TSP to a similar problem called the Chinese postman problem which allows a polynomial time solution. However, a slight modification of the model leads to an NP-hard problem again [9,10]. Another very basic and extensively studied problem is the graph colouring problem that is of great theoretical interest in mathematics, but also has interesting applications. In some sense, a graph colouring problem has to be solved in any scheduling problem. Furthermore, most channel assignment problems can be regarded as a generalization of the basic graph colouring problem [34].

In computational complexity theory, the complexity class NP-complete (NPC) is a class of decision problems. The corresponding optimization problems are NP-hard. Although any given solution to an NP-complete or NP-hard problem can be verified quickly (in polynomial time), there is no known efficient way to find a solution. As a consequence, determining whether or not it is possible to solve these problems quickly, called the P versus NP problem, is one of the principal unsolved problems in theoretical computer science today. It is included in the list of seven millennium problems, and a million dollars prize is offered for a solution [31]. NP-hard problems are often addressed by using heuristic methods and approximation algorithms. Combinatorial optimization problems can be viewed as searching for the best element of some set of discrete items; therefore, in principle, any sort of search algorithm or metaheuristic can be used to solve them. However, generic search algorithms are not guaranteed to find an optimal solution, nor are they guaranteed to run quickly (in polynomial time). Assuming that the conjecture $P \neq NP$ is true, there is no efficient (i.e. polynomial time) algorithm for any NP-hard problem. Either the algorithm has superpolynomial time complexity or its results are not expected to be exact. In other words, we cannot expect that the result of the optimization algorithm will always be the exact optimum. Such algorithms are called heuristic algorithms.

III. HEURISTICS AND METAHEURISTICS

A metaheuristic is a higher-level procedure or heuristic that may provide a sufficiently good solution to an optimization problem, especially with incomplete or imperfect information or limited computation capacity. Metaheuristics in contrast to heuristics often make few 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. This means that for some reasons we strongly hope that the solution will be of good quality, but in general we have no approximation guarantee. Many books and survey papers have been published on the subject, for example [21].

Most studies on metaheuristics are experimental, describing empirical results based on computer experiments with the algorithms. But some formal theoretical results are also available, often on convergence and the possibility of finding the global optimum. Here we will mention some examples of both experimental work and a theoretical result based on author's experience. The selection of examples is therefore biased, and many relevant papers are not mentioned as we have no intention to provide a comprehensive survey.

Metaheuristics are used for combinatorial optimization in which an optimal solution is sought over a discrete search-space. An example is the travelling salesman problem where the search-space of candidate solutions grows faster than exponentially as the size of the problem increases, which makes an exhaustive search for the optimal solution infeasible. In the search space of feasible solutions the solutions with extremal values of the goal functions are to be found.

Perhaps the most natural and conceptually simple metaheuristics is the local search. In order to speak about local search, a topology in the search space is introduced, usually via definition of neighbourhood 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, best neighbour (or steepest descent) and iterative improvement. When a maximal solution is searched, iterative improvement is sometimes called Hill Climbing. As the names indicate, starting from initial feasible solution, iterative improvement generates a random neighbour, and moves to the new solution only if it improves the value of the goal function. The procedure stops when there has been no improvement for sufficiently long time. On the other hand, the best neighbour heuristics considers all neighbours and moves to the new solution with the best value of the goal function among the neighbours. If there is no better neighbour, the current solution is clearly local optima. Note that given a particular optimization problem, different neighbourhood structures can be defined giving rise to different local search heuristics. An elegant idea is switch among various neighbourhoods; see for example [15].

In fact, many metaheuristics can be seen as variations or improvement of the local search [1]. Popular examples that can be seen as variations of local search include simulated annealing, taboo search, iterated local search, variable neighbourhood search, and GRASP (Greedy Randomized Adaptive Search Procedure). A seemingly obvious drawback of the local search is its complete lack of memory. Namely, in the basic version and some of the variants, only the best solution is remembered, and all other information that may have been obtained during the computation is lost. An exception is the taboo search that successfully introduces a short time memory. Therefore, it is natural to develop metaheuristics using search strategies that have a learning component added to the search. Metaheuristics motivated by the idea of learning from the past searches include ant colony optimization, evolutionary computation, and genetic algorithms, to name just a few. It is however a good question in each particular case whether learning does indeed mean an improvement. Namely, a successful heuristic search must have both enough intensification and diversification. Indeed, too intensive learning may result in poor diversification with negative effects to performance of the heuristics [23].

The second important issue that may have essential impact on the success of the multistart local search based optimization is the selection of initial solution. Sometimes a solution is given, for example when we have an already known practical solution. 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 reasonably good starting solutions that in the same time are randomly generated thus assisting the multistart algorithm diversification.

IV. SOLUTIONS BY GREEDY CONSTRUCTIONS

Intuitive explanation of an NP-hard problem may be to roughly say that the problem is likely to be intractable when the greedy construction does not work. Indeed for many problems we have theorems saying that some greedy algorithms will provide nonoptimal solutions. A well known example is the nearest neighbour heuristics for TSP for which we know that can provide arbitrary bad solutions on some artificially created instances. On the other hand, a randomized greedy construction heuristics can be rather useful [2,16,25,26]. We will discuss a couple of examples in more detail below.

Example. TSP HEURISTICS BASED ON GREEDY CONSTRUCTIONS. The traveling salesman problem (TSP) is one of the most studied problems in combinatorial optimization [12]. The TSP is simply stated, has practical applications and is a representative of a large class of important scientific and engineering problems. An instance of TSP is given by a distance matrix $D = (d_{ij})$ of dimension n x n; where d_{ij} represents the weight of the arc from city i to city j in $n = 1,...,n$. If $d_{ij} = d_{ji}$ for every pair i and j then the TSP is *symmetric*, otherwise it is *asymmetric*, (ATSP). TSP is an example of a NP-hard problem. It is therefore reasonable to design algorithms which find near-optimal solutions. Several hundreds of papers were published on TSP and probably every approach for attacking NP-hard optimization problems has also been tested or has even been originally formulated for TSP. Here we consider the heuristics which is based on the well-known arbitrary insertion procedure. It is called Randomized Arbitrary Insertion (RAI), see the pseudocode below. This procedure was not payed too much attention, maybe because of the known worst case performance. However, in the experiment reported in [2] we got surprisingly good results within acceptable computation times. The algorithm was tested on all ATSP instances of the TSPLIB library [32], which were available at the time of the experiment. The main idea of the heuristic is based on the arbitrary insertion algorithm, a relaxation of the cheapest insertion algorithm. The solutions are further improved by a local optimization phase.

Algorithm RAI (Randomized Arbitrary Insertion):

```
A. TOUR CONSTRUCTION
1. Start with a tour consisting of a given vertex and a self-loop.
2. While there are vertices not on the tour do repeat
        2.1. Randomly choose a vertex not on the tour.
        2.2. Insert this vertex between two consecutive vertices on the tour in the cheapest way.

B. TOUR IMPROVEMENT
3. Remember the tour solution, say S.
4. Repeat N times
        4.1. Randomly choose i and j ($1 \le i < j \le n$).
        4.2. Remove vertices i, i+1,…, j, and connect vertex i with vertex j + 1.
        4.3. While there are vertices not on the tour do repeat
                4.3.1. Randomly choose a vertex v not on the tour.
                4.3.2. Insert v between two consecutive vertices on the tour in the cheapest way.
        4.4. Compare current solution with the solution S. Keep the better one.
```

First two steps construct an initial solution. In the second part, in steps 3 and 4, the current solution is improved. This can be seen as a fixed number of moves of iterative improvement, where the neighbourhood is defined in a slightly nonstandard way. However, observe that the neighbourhood is consistent with the construction phase.

As TSP is extensively studied problem, it is easy to compare a new heuristics with good competitors. Instances from TSPLIB, the library of TSP instances are natural dataset for testing. In the experiment [2] the algorithm was run on all 27 asymmetric instances in TSPLIB available at the time. All solutions were within 3% from the optimal value, and in average over 50 runs, all instances were solved within 0.5% from optimum!

In addition, and quite surprisingly, comparison of the heuristic with Farthest Insertion (far), and Farthest Insertion followed by OR-opt [12], which is widely recognized as a very good heuristics of this type, made it possible to conclude that the fast and simple heuristics RAI performs remarkably well and is competitive both in terms of solution quality and execution times with the best heuristics proposed in the literature. In the table below, farthest insertion was repeated n times, each time another vertex was used as initial tour. For the algorithm RAI, the average results of 50 runs are given. Within comparable execution times the solutions obtained by RAI algorithm were better with only one exception, the instance ftv33. For more details see [2].

Table 1: Comparison our algorithm with farthest insertion

	ftv33		ftv64		ftv170		rbg323		rbg443	
Algorithm	best solution	time [s]	best solution	time [s]	best solution	time [s]	best solution	time [s]	best solution	time [s]
far	1298	0.123	2043	1.54	3297	74.38	1672	1204.6	3163	6211.2
far+OR	1286	0.596	1912	4.50	2957	286.6	1359	4963.1	2733	17573
RAI	1288.16	0.455	1861.62	5.48	2832.74	276.3	1348.95	3888.3	2720.7	14342

Figure below shows how the quality of the solutions (minimum, maximum and average) is improving with time on a typical instance.

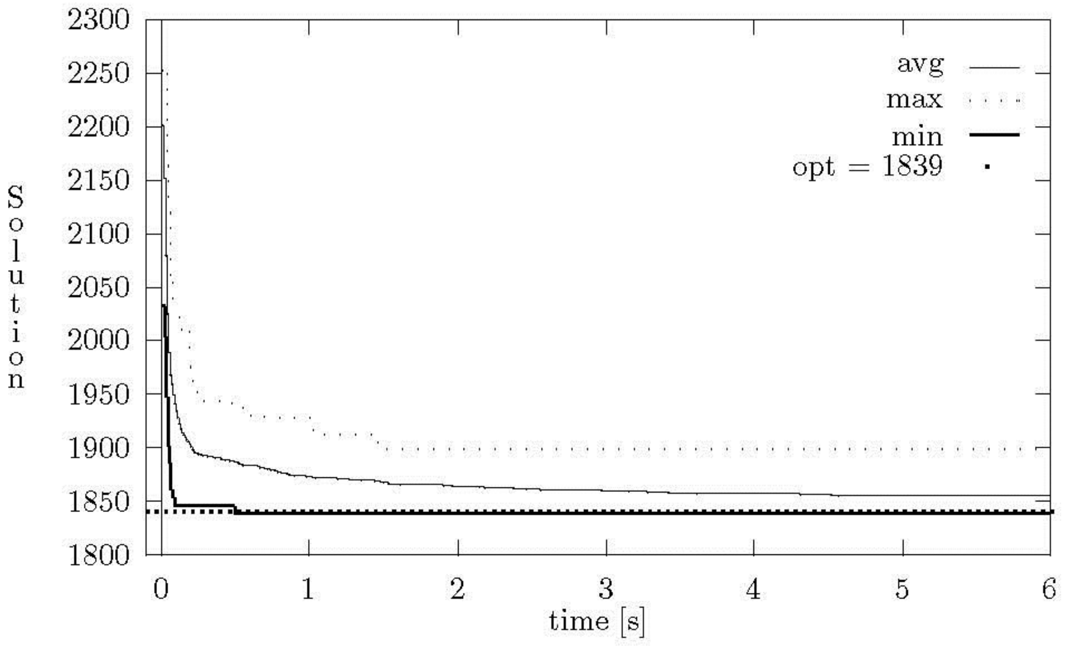

Figure 1. Solution vs. time on ftv64 instance

The experiment on TSP (and ATSP) was inspired by promising results that we got with insertion and optimization approach on PTSP [26], a probabilistic generalization of TSP.

Logistics & Sustainable Transport
Vol. 6, No. 1, November 2015, 1–10
doi: 10.1515/jlst-2015-0006

Example. RCPSP HEURISTICS BASED ON GREEDY CONSTRUCTIONS. The resource-constrained project scheduling problem (RCPSP) can be stated as follows [16]. Given are n activities $a_1,\dots,a_n$ and K renewable resources. A constant amount R_k of units of resource k is available at any time. Activity a_i must be processed for p_i time units; preemption is not allowed. During this time period a constant amount of r_{ik} units of resource k is occupied. All the values R_k, p_i, and r_{ik} are non-negative integers. Furthermore, there are precedence relations defined between activities. That is, we are given an undirected graph G = (V,E) with V = {1,…,n} such that if (i,j) ∈ E then activity j cannot start before the end of activity i. The objective is to determine starting times s_i for the activities a_i, i = 1,…,n in such a way that: at each time t the total resource demand is less than or equal to the resource availability for each resource, the precedence constraints are fulfilled and, the makespan $C_{max} = \max \{c_i \mid i = 1,\dots,n\}$, where $c_i = s_i + p_i$, is minimized. As a generalization of the job-shop scheduling problem the RCPSP is NP-hard.

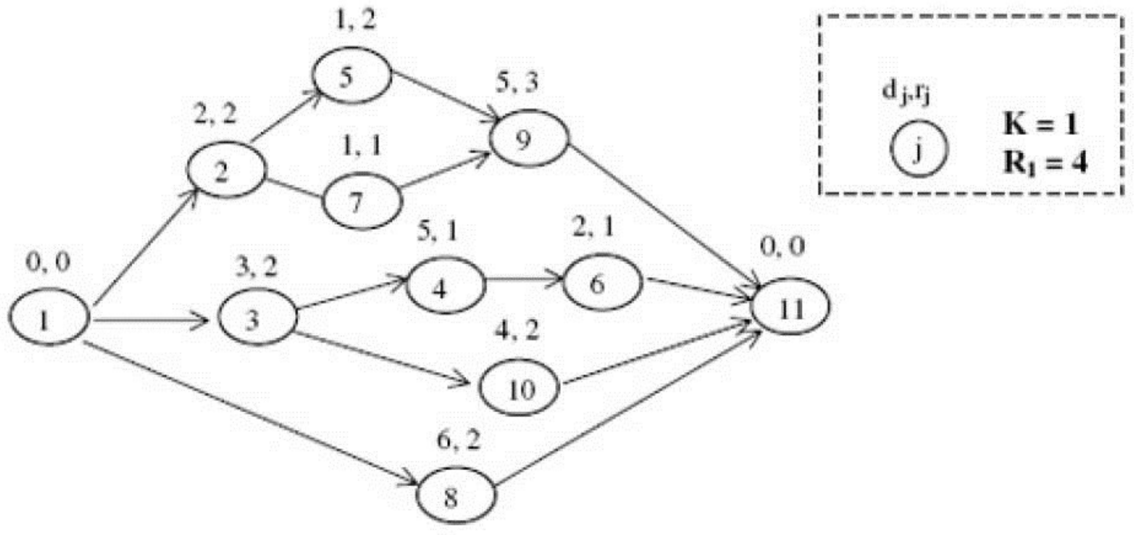

Figure 2. A RCPSP instance. One resource (K=1) with 4 units available (R_1=4).

Let us illustrate the above definitions with the example on Figure 2. There are eleven activities that are partially ordered. The numbers give the length of each activity and the units of resource to be occupied. A feasible schedule can be represented in different ways. Besides the assignment of starting times, one can also give an ordered list of activities. This list has to be precedence feasible, i.e. each activity has all its network predecessors as list predecessors. Given the activities can be scheduled in the order of the list at the earlier precedence and resource-feasible start time. Clearly, there always exists a list that will generate an optimal schedule. Given the network depicted above, some of the (feasible) activity lists are: λ = (1; 2; 3; 5; 7; 9; 8; 4; 6; 10; 11) μ = (1; 2; 3; 7; 10; 4; 8; 5; 9; 6; 11) and δ = (1; 2; 3; 7; 4; 10; 8; 5; 6; 9; 11). For example, the makespan of μ is 18. The activity list gives rise to a schedule depicted below.

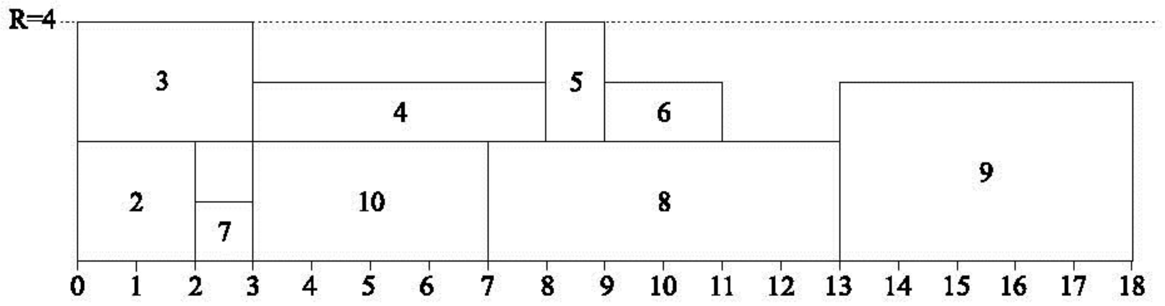

Figure 3. The schedule from activity list μ, makespan is 18.

In [16], a constructive heuristics followed by local search using the same type neighbourhood was successfully applied. Again, the initial solution is constructed from scratch in a stepwise extension of the partial schedule. In each step, an activity is randomly selected from the set of unscheduled activities taking into account the precedence relationships of the activities and the availability of the resources. This approach is thoroughly investigated in [6]. The main idea of the neighbourhood that is called HC(RaR) (Hill Climbing, Remove and Reinsert) is to remove a fixed number m of activities from the schedule and insert them back into the schedule, where m is a parameter of the method. For details, see [16]. It has been shown that the heuristics is competitive with the best know heuristics for the problem. In the table below, the best ten algorithms reported at the time are compared (some further algorithms are reported in [7]. The instances are taken from library PSPLIB [33]. The first column gives the method used, the second provides names of the authors of the method and the last column gives the average deviation from the critical path lower bound.

Table 2: Comparison algorithms for RCPSP

Method	Author	avg. dev.(%)
Scatter search	Debels et al	10.71
GA-hybrid	Valls et al.	10.73
GA, TS-path relinking	Kochetov and Stolyar	10.74
GA-FBI	Valls et al.	10.74
GA-forw.-backw., FBI	Alcaraz et al.	10.84
HC(RaR)	**Pesek, Schaerf, Žerovnik**	**11.10**
GA-self-adapting	Hartmann	11.21
GA-activity list	Hartmann	11.23
Sampling-LFT, FBI	Tormos and Lova	11.36
TS-activity list	Nonobe and Ibaraki	11.58

V. LOCAL SEARCH VERSUS SIMULATED ANNEALING

Simulated annealing (SA) [11] can be understood as a random relaxation of the iterative improvement algorithm [1]. It was a very popular heuristics in the 90's and it is still among the frequently used relaxations of iterative improvement. Roughly speaking, given any iterative improvement with a given neighbourhood structure, the simulated annealing heuristics can be defined by the following relaxation of the acceptance rule. If the neighbour is not better, then accept the move with certain probability. The acceptance probability depends on the parameter usually called the temperature. Sometimes very good solutions are found by SA, but it is also well known that SA can be extremely time consuming. Furthermore, the best known convergence results hold only if the so-called cooling schedule is very slow [5,14]. On the other hand, it is known for some time that at least asymptotically and with a usual implementation, where rejected moves are explicitly counted, the probability of success of simulated annealing is worse than the probability of success of as simple a procedure as multistarts of the iterative improvement algorithm [4]. Detailed analysis of various temperature schedules are given for example in [3,19,28]. Interesting enough, among various temperature schedules, the constant temperature schedule for simulated annealing was found competitive or at least worth consideration. Can a schedule be more simple than the constant one?! We also have some very positive experience with a SA-like algorithm for graph colouring [17,27], namely very fast convergence of the algorithm on an interval of good temperatures [20]. It should be noted that this graph colouring algorithm is quite robust and can be applied to several generalizations including channel assignment problems [22].

Another example is a successful search for independent sets in certain graph products using simulated annealing at constant temperature [24] that has lead to the best upper bound for the Shannon capacity of C_7 which was not improved until today. The Shannon capacity of the 7-cycle is one of the long lasting unsolved problems and until now only lower and upper bounds were found and are gradually improved. It may be interesting to note that the capacity of C_5 was studied already by Shannon [18] in 1956, and was determined only in 1979 by Lovász [13]. In other words, simple variant of multistart local search is so far the winner in this example. More precisely, the local search is a simulated annealing at fixed temperature, where the temperature used was established by a very fast and simple preprocessing. For further discussion and references on fixed temperature schedules see [29].

Similarly to some other metaheuristics, simulated annealing was very popular due to motivation from natural phenomena. There was a substantial effort to prove convergence results, and there are hundreds of reported applications. An important issue in designing a successful implementation of SA is to choose a lucky temperature schedule. Is it worth effort to design a complicated heuristics with parameters difficult to understand ? On the other hand, we can understand SA as a relaxation of local search, and using a multistart version gives the convergence trivially. Furthermore, constant temperature schedule (the simplest schedule possible!) may be a promising choice. In our view, SA is an interesting randomized relaxation of iterative improvement. In many examples, already the basic iterative improvement (with multistart) provides a successful method when the neighbourhood is chosen with some care and some luck.

VI. CONCLUSIONS

We have recalled two results showing that a natural greedy construction followed by iterative improvement can be a very competitive heuristics. The examples shared the property that the greedy construction can naturally be randomized, thus enabling sufficient diversification of the initial solutions. This approach is only slightly different from another successful heuristics, GRASP [35].

We have also briefly discussed the simulated annealing metaheuristics, its relation to iterative improvement and the importance of the temperature schedule. Both theory and practical examples indicate that simple temperature schedules may be among the competitive ones.

We have thus provided some arguments for the hypothesis that when solving hard optimization problems, simpler is often also better. More precisely, better here means that solutions competitive in quality can be found in reasonable time, but also that implementation and tuning of the algorithms can be done more easily. Last but not least, we should take into account that the complicated metaheuristics usually have a number of parameters that have to be tuned to obtain the best results. The task of tuning the parameters however may itself be a very hard one, and can also be very time consuming!

Finally, we wish to add that there are more recent examples supporting the ideas given above. An algorithm based on RaR has recently been applied to job shop scheduling with success [36], and several local search heuristics to a problem related to design of optical systems based on LED technology [37,38]. In both applications, the heuristics used have been very competitive in comparison with genetic algorithms.

In conclusion, it is clear that the question and the suggested answer provided in the title are very general. Such a general question is difficult to be stated in a formal way, and thus very difficult to provide an answer that can be proved formally. However, we do not claim that the answer suggested is valid for all cases.

ACKNOWLEDGEMENT

The author wishes to thank to the anonymous reviewer for careful reading of the manuscript and for constructive comments.

REFERENCES

1. E. H. L. Aarts and J. K. Lenstra, Local Search Algorithms, John Wiley & Sons, Chichester, 1997.

2. J. Brest and J. Žerovnik, "An approximation algorithm for the asymmetric traveling salesman problem," *Ricerca operativa*, vol. 28, pp. 59–67, 1999.
3. H. Cohn and M. Fielding, "Simulated annealing: searching for an optimal temperature schedule," *SIAM J. Optim.*, Vol. 9, pp. 779–802, 1999.
4. A. G. Ferreira and J. Žerovnik, "Bounding the probability of success of stochastic methods for global optimization," *Computers & mathematics with applications*, vol. 25, pp. 1–8, 1993.
5. B. Hajek and G. Sasaki, "Simulated annealing - to cool or not," *System Control Lett.*, vol.12, pp. 443–447, 1989.
6. R. Kolisch and S. Hartmann, "Heuristic algorithms for solving the resource-constrained project scheduling problem - classification and computational analysis," in *Handbook on recent advances in project scheduling*, J. Weglarz, Ed., Kluwer, 1999, pp. 147–178.
7. R. Kolisch and S. Hartmann, "Experimental investigation of heuristics for resource-constrained project scheduling: An update," European Journal of Operational Research, vol. 174, pp. 23–37, 2006.
8. B. Korte and J. Vygen, Combinatorial optimization: Theory and algorithms, Springer, Berlin, 2002.
9. T. Kramberger, G. Štrubelj and J. Žerovnik, "Chinese postman problem with priority nodes, " *Foundations of computing and decision sciences*, 2009, vol. 34, no. 4, pp. 233–264, 2009.
10. T. Kramberger and J. Žerovnik, "Priority constrained Chinese postman problem", *Logistics & sustainable transport*, vol. 1, pp. 21–35 , 2008.
11. P.J. van Laarhoven and E.H. Aarts, Simulated Annealing: Theory and Applications, Kluwer, Dordrecht, 1987.
12. E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, and D.B Shmoys, The traveling salesman problem, Wiley & Sons, Chichester, 1985.
13. L. Lovasz, "On the Shannon capacity of a graph, " *IEEE Trans. Inform. Theory*, vol. 25, 1–7, 1979.
14. D. Mitra, F. Romeo and A. Sangiovanni-Vincentelli, "Convergence and finite time behavior of simulated annealing, " *Adv. in Appl. Probab.*, vol. 18, pp. 747–771, 1986.
15. N. Mladenović and P. Hansen: "Variable neighborhood search, " *Computers & OR*, vol. 24, pp. 1097–1100, 1997.
16. I. Pesek, A. Schaerf, and J. Žerovnik, "Hybrid local search techniques for the resource-constrained project scheduling problem," *Lecture notes in computer science*, vol. 4771, pp. 57–68, 2007.
17. A. Petford and D. Welsh, "A randomised 3-colouring algorithm, " *Discrete Mathematics*, vol. 74, pp. 253–261, 1989.
18. C. E. Shannon, "The zero-error capacity of a noisy channel, " *IRE Trans. Inform. Theory*, vol. 2, pp. 8–19, 1956.
19. J. Shawe-Taylor and J. Žerovnik, "Analysis of the mean field annealing algorithm for graph colouring," *Journal of artificial neural networks*, vol. 2, pp. 329–340, 1995.
20. J. Shawe-Taylor and J. Žerovnik, "Adapting temperature for some randomized local search algorithms," in: Advances in scientific computing, computational intelligence and applications, N. Mastorakis, Ed. WSES Press, 2001, pp. 82–87.
21. E.-G. Talbi, Metaheuristics: From Design to Implementation, John Wiley & Sons, 2009.
22. S. Ubeda and J. Žerovnik, "A randomized algorithm for a channel assignment problem," *Speedup*, vol. 11, pp. 14-19, 1997.
23. A. Vesel and J. Žerovnik, "How well can ants colour graphs?" *CIT*, vol. 8, pp. 131–136, 2000.
24. A. Vesel and J. Žerovnik, "Improved lower bound on the Shannon capacity of C_7," *Information processing letters*, 2002, vol. 81, pp. 277–282, 2002.
25. G. Žerovnik and J. Žerovnik, "Constructive heuristics for the canister filling problem," *Central European Journal of Operations Research*, 2011, vol. 19, pp. 371–389, 2011.
26. J. Žerovnik, "A heuristics for the probabilistic traveling salesman problem," in *Symposium on Operation Research '95*, V. Rupnik and M. Bogataj, Eds. Ljubljana: Slovenian Society Informatika, 1995, pp. 165–172.
27. J. Žerovnik, "A randomized algorithm for k-colorability," *Discrete Mathematics*, vol. 131, pp. 379–393, 1994.
28. J. Žerovnik, "On temperature schedules for generalized Boltzmann machine," *Neural network world*, vol. 10, pp. 495–503, 2000.
29. J. Žerovnik, "Simulated annealing type metaheuristics - to cool or not to cool," in 7th International Symposium on Operational Research in Slovenia, L. Zadnik Stirn, M. Bastič and S. Drobne, Eds. Ljubljana: Slovenian Society Informatika, 2003, 6 pp.
30. http://en.wikipedia.org/wiki/Combinatorial_optimization
31. http://www.claymath.org/millenium-problems/p-vs-np-problem
32. http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/
33. http://129.187.106.231/psplib/
34. http://fap.zib.de/index.php
35. http://en.wikipedia.org/wiki/Greedy_randomized_adaptive_search_procedure
36. H. Zupan, N. Herakovič, and J. Žerovnik, "A hybrid metaheuristic for job-shop scheduling with machine and sequence-dependent setup times," in 13th International Symposium on Operational Research in Slovenia, L. Zadnik Stirn, M. et al., Eds. Ljubljana: Slovenian Society Informatika, 2015, pp. 129-134.
37. D. Kaljun, and J. Žerovnik, "On modelling of spatial light distribution of LED with attached secondary optics," in 13th International Symposium on Operational Research in Slovenia, L. Zadnik Stirn, M. et al., Eds. Ljubljana: Slovenian Society Informatika, 2015 , pp. 363-368.

38. D. Kaljun, D. Rupnik Poklukar, and J. Žerovnik, Janez. "Heuristics for optimization of LED spatial light distribution model," *Informatica*, vol. 39, pp. 147-159, 2015. http://www.informatica.si/index.php/informatica/article/view/831/625.

AUTHOR

J. Žerovnik is with the Faculty of Mechanical Engineering, University of Ljubljana, Aškerčeva 6, Ljubljana, Slovenia and the Institute of Mathematics, Physics and Mechanics, Jadranska 19, Ljubljana, Slovenia (e-mail: janez.zerovnik@ fs.uni-lj.si).