Classification of Metaheuristics and Design of Experiments for the Analysis of Components

Tech. Rep. AIDA-01-05

Mauro Birattari, Luis Paquete,
Thomas Stützle, and Klaus Varrentrapp

Intellektik
Darmstadt University of Technology
Darmstadt, Germany

November 2001

Abstract

This report discusses two different approaches to the description of metaheuristics. On one hand, we propose a number of different high-level criteria according to which metaheuristics can be described and classified. On the other hand, we discuss some method of design of experiments for studying the contribution and the relative importance of the different components of a metaheuristic. We maintain that, in order to be effective and complete, an analysis of metaheuristics must take into account both levels. In particular, such a two-fold investigation is necessary for coping with the fact that all metaheuristics, though described as clearly defined methods based on some intellectually appealing principles, when implemented in practice, they always appear in some hybrid form. Accordingly, with an high-level description, we wish to grasp the underlying principle, while by adopting statistical techniques of design of experiments we wish to ponderate the contribution given to performance by each components in its possible instantiations.

1 Introduction

We see two main approaches in the analysis of Metaheuristics. The firsts looks at metaheuristics in their "pure state". The second takes into account the fact that realistic implementations of metaheuristics always include "dirty tricks".

When we look at pure metaheuristics, a clear-cut classification is possible of the underlying principles and mechanisms. In this empyrean, metaheuristics can be easily disassembled into components, and alternative ways of reassembling these components into possibly hybrid new metaheuristics can be devised.

The praxis of metaheuristics is somehow different. Real world implementations of metaheuristics usually exploit a large amount of heterogeneous ideas and "tricks". It can be argued that a successful metaheuristic is the one that lends itself to hybridization and flexible adaptation that are easyly implementable. It is fundamental therefore to provide guidelines for the description and the analysis of real world implementations of metaheuristics. Sharp categories appears not to be suited for such a description, and a language that gives an account of all possible "tones of gray" seems to be needed.

We maintain that the two approaches, component-level and whole-level, are both important: They play a different role and must be both present in an analysis.

As far as the whole-level is concerned, an analysis of different metaheuristics can be obtained through competitive tests over a class of instances that are commonly considered in published material. Although this kind of analysis is frequently performed, it cannot be considered as the only source of information on the metaheuristics of interest. The limitations of competitive testing and the associated risks are well known (Hooker, 1996). Nevertheless, this kind of analysis is often justified on at least two different grounds. First, performance is typically the criterion of primary importance when comparing metaheuristic. Second, this analysis is easy to be performed and can provide initial clues on the behavior of the different metaheuristics.

As an alternative to competitive testing, more systematic and "scientific" testing methodology are available. In particular, practical and valuable information can be extracted at the component-level through techniques of *design of experiments* and statistical analysis.

To date, extensive testing of different metaheuristics implemented by the Metaheuristics Network has been performed. Results are available for what concerns QAP and MAX-SAT. Future analysis will aim at assessing the relative contribution of the various components and possibly the role that components play across different metaheuristics.

The rest of this report is structured as follow. Taking inspiration from Stützle (1998), Section 2 proposes a high-level analysis of metaheuristics. Different criteria of classification are presented and discussed.

Section 3 introduces some elements of design of experiments and gives a preliminary description of a methodology to be applied to testing of metaheuristics. Further, Section 3 provides some useful references to the relevant literature.

2 Classification of metaheuristics

A metaheuristic will be successful on a given optimization problem if it can provide a balance between the exploitation of the accumulated search experience and the exploration of the search space to identify regions with high quality solutions in a problem specific, near optimal way. The main difference between the existing metaheuristics concerns the particular way in which they try to achieve this balance. The different metaheuristic approaches can be characterized by different aspects concerning the search path they follow or how memory is exploited. In this section, we discuss these aspects according to some general criteria which may be used to classify the presented algorithms. For a more formal classification of local search algorithms based on an abstract algorithmic skeleton we refer to (Vaessens et al., 1995).

Trajectory methods vs. discontinuous methods An important distinction between different metaheuristics is whether they follow one single search trajectory corresponding to a closed walk on the neighborhood graph or whether larger jumps in the neighborhood graph are allowed. Of the presented metaheuristics, simulated annealing and tabu search are typical examples of trajectory methods. These methods usually allow moves to worse solutions to be able to escape from local minima. Also local search algorithms which perform more complex transitions which are composed of simpler moves may be interpreted as trajectory methods. Such algorithms are, for example, variable depth search algorithms like the Lin-Kernighan heuristic for the TSP (Lin and Kernighan, 1973) and algorithms based on ejection chains (Glover, 1996). In ant colony optimization, iterated local search, genetic algorithms, and grasp starting points for a subsequent local search are generated. This is done by constructing solutions with ants, modifications to previously visited locally optimal solutions, applications of genetic operators, and randomized greedy construction heuristics, respectively. The generation of starting solutions corresponds to *jumps* in the search space; these algorithms, in general, follow a discontinuous walk with respect to the neighborhood graph used in the local search.

Population-based vs. single-point search Related to the distinction between trajectory methods and discontinuous walk methods is the use of a population of search points or the use of one single search point. In the latter case only one single solution is manipulated at each iteration of the algorithm. Tabu search, simulated annealing, iterated local search, and grasp are such single-point search methods. On the contrary, in ACO algorithms and genetic algorithms, a population of ants or individuals, respectively, is used. (Note that population-based methods are typically discontinuous.) In ant colony optimization a colony of ants is used to construct solutions guided by the pheromone trails and a heuristic function and in genetic algorithms the population is modified using the genetic operators. Using a population-based algorithm provides a convenient way for the exploration of the search space. Yet, the final performance depends strongly on the way the population is manipulated.

Memory usage vs. memoryless methods Another possible characteristic of metaheuristics is the use of the search experience (memory, in the widest sense) to influence the future search direction. Memory is *explicitly* used in tabu search. Short term memory is used to forbid revisiting recently found solutions and to avoid cycling, while long term memory is used for diversification and intensification features. In ant colony optimization an indirect kind of *adaptive* memory of previously visited solutions is kept via the pheromone trail matrix which is used to influence the construction of new solutions. Also, the population of the genetic algorithm could be interpreted as a kind of memory of the recent search experience. Recently, the term *adaptive memory programming* (Taillard et al., 1998) has been coined to refer to algorithms that use some kind of memory and to identify common features among them. Also iterated local search, in a widest sense could be classified as an adaptive memory programming algorithm, although only very poor use of the recent search experience is made in ILS algorithms like choosing the best solution found so far for the modification step. On the contrary, simulated annealing and grasp do not use memory functions to influence the future search direction and therefore are memoryless algorithms.

One vs. various neighborhood structures Most local search algorithms are based on one single neighborhood structure which defines the type of allowed moves. This is the case for simulated annealing and tabu search. Iterated local search algorithms typically use at least two different neighborhood structures $\mathcal{N}$ and $\mathcal{N}'$. The local search starts with neighborhood $\mathcal{N}$ until a local optimum is reached and in such a situation a kick-move is applied to catapult the search to another point. In fact, the kick-moves can be interpreted as moves in a secondary neighborhood $\mathcal{N}'$. For the subsequent local search again the primary neighborhood $\mathcal{N}$ is used. Often, an appropriate strength of the kick-move is not known or may depend on the search space region. Therefore, it may be advantageous to choose several neighborhoods $\mathcal{N}_1, \ldots, \mathcal{N}_k$ of different size for the kick-moves. Simple variable neighborhood search (VNS) introduces this idea by systematically changing the neighborhood (Mladenović and Hansen, 1997). The mutation operator in genetic algorithms has the same effect as the kick-move in ILS and therefore may also be interpreted as a change in the neighborhood during the local search. Applications of the crossover operator have been interpreted as moves in hyper-neighborhoods (Vaessens et al., 1995), in which a cluster of solutions – in genetic algorithms these clusters are of size two – is used to generate new solutions. On the other side, the solution construction process in ant colony optimization and grasp is not based on a specific neighborhood structure. Nevertheless, one could interpret the construction process used in ACO and grasp as a kind of local search, but this interpretation does not reflect the basic algorithmic idea of these approaches.

Dynamic vs. static objective function Some algorithms modify the evaluation of the single search states during the run of the algorithm. One particular example, which has not been discussed before, is the breakout method (Morris, 1993) proposed for the solution of satisfiability and graph coloring problems. The basic idea is to introduce penalties for the inclusion of certain solution attributes which modify the objective function. Based on the breakout method, *guided local search* (Voudouris and Tsang, 1995) has been proposed and applied to combinatorial optimization problems like the TSP. Also tabu search may be interpreted as using a dynamic objective function, as some points in the search space are forbidden, corresponding to infinitely high objective function values. Yet, all the other algorithms introduced so far use a static objective function.

Nature-inspired vs. non-nature inspiration A minor point for the classification of metaheuristics is to take into account their original source of inspiration. Many methods are actually inspired by naturally occurring phenomena. The algorithmic approaches try to take advantage of these phenomena for the efficient solution of combinatorial optimization problems. Among the presented methods, ant colony optimization, simulated annealing, and genetic algorithms belong to these nature-inspired algorithms. The others, tabu search, iterated local search, and grasp have been inspired more by considerations on the efficient solution of combinatorial problems.

In Table 1 we summarize the discussion of the various methods according to these criteria. We do not claim that *all* implementations of these algorithms correspond to this classification, but it rather gives an indication of the particular characteristics of these methods in their "standard" use.

3 Statistical analysis of factors

3.1 Design of Experiments

Before going into the details of design of experiments, the terminology used in this research field must be introduced. In particular, we need to associate the terms used in design of experiments with concepts and objects encountered when dealing with metaheuristics.

When refering to *factors*, we are actually talking about the high-level components that fully characterize a metaheuristic. However, if we intend to study

Table 1: Summary of the characteristics discussed in this section. $\surd$ means that the feature is present, $\exists$ that this feature is partially present and $\neg$ that the feature does not appear.

Feature	SA	TS	GA	ACO	ILS	grasp	GLS
Trajectory	$\surd$	$\surd$	$\neg$	$\neg$	$\neg$	$\neg$	$\surd$
Population	$\neg$	$\neg$	$\surd$	$\surd$	$\neg$	$\neg$	$\neg$
Memory	$\neg$	$\surd$	$\exists$	$\surd$	$\exists$	$\neg$	$\surd$
Multiple neighborhoods	$\neg$	$\neg$	$\exists$	$\neg$	$\surd$	$\neg$	$\neg$
Dynamic $f(x)$	$\neg$	$\exists$	$\neg$	$\neg$	$\neg$	$\neg$	$\surd$
Nature-inspired	$\surd$	$\neg$	$\surd$	$\surd$	$\neg$	$\neg$	$\neg$

also the behavior over different classes of instances of the same problem, where each instance is defined by a unique combination of certain values of its features, also these features can be defined as *factors* (Rardin and Uzsoy, 2001). For instance, in a computational experiment dealing with the graph coloring problem, the number of nodes and edge density, that is, features that fully characterize a certain graph instance, are considered also as *factors*, since they can also influence the behavior of the metaheuristic.

With the term *levels* we refer to the possible values or options that a given *factor* may take. Levels could represent parameters (length of tabu list, population size) or methods (type of perturbation, mutation operator), or even combination of these (length of tabu list of method robust tabu search). Factors are usually run at only two levels: "high" and "low". Although it is possible to observe the effect of the factor, the results depend on what levels are chosen. To determine whether an effect is linear or nonlinear, at least three levels should be included.

In the design of experiments, both *randomization* and *blocking* are relevant aspects. By *randomization* we mean that all factors not explicitly controlled should be set randomly to prevent systematic and personal bias (Dean and Voss, 1999). Considering the previous example, if we generate several graph instances by randomly defining edge density and number of nodes, *randomization* strictly says that different instances should be randomly chosen for the considered set of combinations of high-level components of a metaheuristic.

One drawback of this design is that it increases the variance in the observed data. In order to reduce such variability, a technique called *blocking* is often adopted over the factors that are not of interest, called nuisance factors. *Blocking* consists in dividing the observations into groups called *blocks* in such a way that the observations in each *block* are collected under similar conditions (Dean and Voss, 1999).

In experimental design of metaheuristics we can simply fix the nuisance factor during the experiment, (although it could limit the conclusions to that particular chosen level), or rather we can implement the *blocking* procedure by defining levels for the nuisance factor and running a set of experiments for each level. However, some adjustments must be considered in the factor analysis.

Most of the current approaches to experimental design use some kind of blocking (Lin, 1980; Hooker and Vinay, 1995; Paquete and Fonseca, 2001; Golden and Stewart, 1985).

Specific types of experimental design can be well suited to computational experiments where the nuisance factors are blocked and there is only one factor under study. The goal of these designs is to study if the main factor is really significant or not; in other terms, if there is a significant change in the response of the several levels of this factor.

Latin Square experimental design uses only one important factor and two nuisance factors. The experiment assume that two nuisance factors are divided in a tabular grid with the property that each row and each column must run only once with each *level* of the *factor* under study.

Graeco-Latin Squares can use three nuisance factors and Hyper-Graeco-Latin Squares can use four. One advantage of these approaches is the reduction of the number of runs. On the other hand, an obvious drawback is the need for the same number of levels between main factor and the others nuisance factors, which is not always possible when dealing with metaheuristics. In any case, even when this could be possible, it would not match with the goal of our analyses.

Most of the research in optimization by means of metaheuristics deals with several important factors and the need emerges of screening out the most important ones. This screening objective can be achieved by factorial design. In this approach, all combination of *all* levels that *all* factors might take need to be considered in the experiments. In Lin (1980) a factorial design is used when two factors are considered (algorithm and problem type). A fixed number of random problems is generated in each combination of two factors. Also in Hooker and Vinay (1995), a factorial design is reported in the experimental study of branching algorithms applied to satisfiability problems.

However, one serious drawback of factorial design is the explosive growing of the number of experiments (McGeoch, 1996; Rardin and Uzsoy, 2001). There are several alternatives which consider a reduced number of combinations, while maintaining the orthogonality of the experiments matrix. Both fractional and Taguchi methods can achieve this by assessing the same effects without running all combinations. These methods need a carefully chosen subset of factors and levels to avoid confounding main effects.

3.2 Factor Analysis

After obtaining the results of the experiments, the research must study the raw data by appropriate statistical tools. Analysis of variance and non-parametric tests, can help to check whether a factor has a significant effect on performance when the remaining factors held constant at any given set of levels (Hooker, 1996). In most of the cases, when considering only the presence of two factors, a two-way table of sample means could help in obtaining some insight about how various factors affect the metaheuristic performance (Rardin and Uzsoy, 2001).

One way of analyzing main effects of the factors under study is the use of Analysis of Variance (ANOVA) (Rardin and Uzsoy, 2001). This statistical procedure assumes that all nonrandom variation is due to differences in mean performance among the several levels of the factors. Assumptions of normality and common error variance are needed to perform this test.

In Hooker and Vinay (1995), multiple regression with dummy variables is used. In this work, the authors consider problem types as an attribute that may affect response variable. They further assume that there is no need to maintain an equal number of problems of each type. These assumptions fit in multiple regression analysis and apply to their own research, since they were dealing with benchmarks.

Non-parametric tests can be used when one is not willing to assume normality. One such test is the permutation test, which only needs the assumption of exchangeability of errors (Good, 1994). This test presents the advantage of not restricting to normality, it is exact, and it allows for the selection of a test statistic to match the alternatives of interest (Good, 1994). In Paquete and Fonseca (2001), for the study of main effects of several methods of a multi-objective metaheuristic, Smirnov distance is used as a test statistic in a permutation test. Reduction of variance and the number of permutations needed was obtained by permuting the elements of the solutions between different levels of the factor of interest, but within the same levels of the other existing factors, in randomization restricted to matching blocks (Good, 1994). However, the computer overload of such test is still highly intensive. Such drawback can be overcomed by approximated procedures based on Monte Carlo sampling.

3.3 Checking the Design of Experiments

Taking as example the checklist described in Dean and Voss (1999), it is possible to define the following methodology to design computational experiments for understanding the behavior of the metaheuristics in combinatorial optimization problems.

Define objectives

Explicitly list all the precise questions to be addressed in the experiment. As far as our research is concerned, the main goal is to study the effect of including different components into the metaheuristic, and to study the behavior of the metaheuristic under different values of its parameters. Other objectives can be formulated, as the one related with screening (Xu et al., 1998) to find the best set of parameters and methods by means of statistical tests and experimental design.

Identify sources of variation

Identify the main factors and their corresponding levels. As discussed above, factors correspond to the high-level methods present in the metaheuristic and its levels are the specific types or amounts of the factor that will actually be used in the experiments. Choose carefully the experimental units which will be used in the experiments. The experiment units can be defined as the instances that will be solved by each combination of the levels of the existing factors and must be representative of the universe of class of problems. Identify also the nuisance factors and in which way can they be defined as blocking factors.

Choose combinations of levels of the factors

Can we have enough computational power to perform a full factorial design? If not, can we use a fractional factorial design? Obviously, these questions depend on the number of factors and levels considered in the previous stages.

Specify the experimental procedure

The units in which the measurements are to be made should be precisely specified. For example, when applying metaheuristics to graph coloring problems, the units could be considered as the number of iterations, number of optimal coloring and/or CPU run-time. In TSP problems, the one that is consider is the length of the tour obtained. Somehow, the units must reflect the objective function of the formalization of the optimization problem. The problem of when to collect data must be addressed, too: Data should be gathered only at the end of iterations, or also at run-time with some specific time interval?

Run a pilot experiment

In many ways, a pilot experiment can be of great help to define the final experiment. Running a pilot experiment can give some insight in the possible problems that must be overcomed, as collecting data and time bounds for each experiment. It can also help to redefine the levels of each factor considered, and the number of observations needed.

Specify the model

The model needs to be specified, that indicates the relationship existing between the response variable and the sources of variation that were previously identified. The techniques for factor analysis will depend of the formulation of this model.

Outline the analysis

Define the statistical analysis to be considered. The statistical hypothesis must be formulated and also the statistical tests. Can we assume normality of the distributions? If not, do we have enough computation time for using permutation tests to determine the exact distributions? In the need of multiple comparisons between data obtained from the experiments, what procedure should be used to control the maximum probability of occurring Type I error? These question are important to define the statistical treatment.

Calculate the number of observation needed
Considering that stochastic algorithms, like metaheuristics, produce different results at each run, a large enough number of observations must be obtained in order to reduce the variability of data. However, a definite method to derive this calculation does not exist. The pilot experiment can be of great importance to define the number of observations needed.

Review the above decisions, revise if necessary
In an analogy to the software design, also here there is the need to restart from the beginning a review and, if necessary, to revise the decisions taken. Most of computational experiments with metaheuristics can take days or weeks. A wrong decision in the design of experiments can cause several delays and unpredictable results.

4 Conclusions and future work

This report starts from the consideration that a two-level analysis and description is fundamental for gaining a satisfactory insight into the behavior and the characteristics of metaheuristics. Still, while a description of a metaheuristic as a whole can be reasonably considered as achieved, much remains to be done at the level of the analysis of components. Metaheuristics as a whole can be described and classified according to different criteria, and can be tested and compare for what their performance is concerned. At the level of an analysis of components, the tools that seem appropriate for gathering and processing information, come from the field of Design of Experiments. Future work will focus on the use of such statistical techniques of experimental design. Beside using such techniques for the analysis of metaheuristics, we will also consider the possibility of adopting them as tools for tuning parameters and, more in general for finding the optimal configuration of a metaheuristic.

Acknowledgments This work was supported by the *Metaheuristics Network,* a Research Training Network funded by the Improving Human Potential Programme of the CEC, grant HPRN-CT-1999-00106. The information provided is the sole responsibility of the authors and does not reflect the Community's opinion. The Community is not responsible for any use that might be made of data appearing in this publication.

References

Dean, A. and Voss, D. (1999). *Design and Analysis of Algorithms.* Springler-Verlag, New York, NY, USA.

Glover, F. (1996). Ejection Chains, Reference Structures and Alternating Path Methods for Traveling Salesman Problems. *Discrete Applied Mathematics*, 65:223–253.

Golden, B. L. and Stewart, W. R. (1985). Empirical analysis of heuristics. In Lawler, E. L., editor, *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*, pages 207–249. John Wiley & Sons, New York, NY, USA.

Good, P. (1994). *Permutation Test: A Practical Guide to Resampling Methods for Testing Hypothesis.* Springler-Verlag, New York, NY, USA.

Hooker, J. (1996). Testing heurists: We have it all wrong. *Journal of Heuristics*, 1:33–42.

Hooker, J. N. and Vinay, V. (1995). Branching rules for satisfiability. *Journal of Automated Reasoning*, 15(3):359–383.

Lin, B. (1980). Controlled experimental design for statistical comparison of integer programming algorithms. *Management Science*, 25:1258–1271.

Lin, S. and Kernighan, B. (1973). An Effective Heuristic Algorithm for the Travelling Salesman Problem. *Operations Research*, 21:498–516.

McGeoch, C. C. (1996). Towards an experimental method for algorithm simulation. *INFORMS Journal of Computing*, 8(1):1–15.

Mladenović, N. and Hansen, P. (1997). Variable Neighborhood Search. *Computers & Operations Research*, 24:1097–1100.

Morris, P. (1993). The Breakout Method for Escaping from Local Minima. In *Proceedings of the 11th Conference on Artificial Intelligence*, pages 40–45. MIT press.

Paquete, L. and Fonseca, C. M. (2001). A study of multiobjective evolutionary algorithms to the examination timetabling problem. In *Proceddings of Metaheuristics International Conference (MIC'01).*

Rardin, R. L. and Uzsoy, R. (2001). Experimental evaluation of heuristic optimization algorithm: A tutorial. *Journal of Heuristics*, 7:261–304.

Stützle, T. (1998). *Local Search Algorithms for Combinatorial Problems—Analysis, Improvements, and New Applications.* PhD thesis, Technische Universität Darmstadt, Darmstadt, Germany.

Taillard, E., Gambardella, L., Gendreau, M., and Potvin, J.-Y. (1998). Adaptive Memory Programming: A Unified View of Metaheuristics. Technical Report IDSIA-19-98, IDSIA, Lugano, Switzerland.

Vaessens, R., Aarts, E., and Lenstra, J. (1995). A Local Search Template (revised version). Technical Report Memorandum COSOR 92-11, Department of Mathematics and Computing Science, Eindhoven.

Voudouris, C. and Tsang, E. (1995). Guided Local Search. Technical Report Technical Report CSM-247, Department of Computer Science, University of Essex, England.

Xu, J., Chiu, S., and Glover, F. (1998). Fine-tuning a tabu search algorithm with statistical tests. *International Transactions in Operational Research*, 5(3):233–244.