Teaching Modern Heuristics in Combinatorial Optimization

The example of a demonstration and research tool employing metaheuristics in scheduling

Martin Josef Geiger
Lehrstuhl für Industriebetriebslehre
Universität Hohenheim
D-70593 Stuttgart
Germany
mjgeiger@uni-hohenheim.de

Abstract. The article describes the proposition and implementation of a demonstration, learning and decision support system for the resolution of combinatorial optimization problems under multiple objectives. The system brings together two key aspects of higher education: research and teaching. It allows the user to define modern metaheuristics and test their resolution behavior on machine scheduling problems. The software may be used by students and researcher with even little knowledge in the mentioned field of research, as the interaction of the user with the system is supported by an extensive graphical user interface. All functions can be easily parameterized, and expensive software licenses are not required. In order to address a large number of users, the system is localizable with little effort. So far, the user interface is available in seven languages.
The software has been honored in Ronneby (Sweden) with the European Academic Software Award 2002, a prize for learning and research software awarded biannually by EKMA, the European Knowledge Media Association (http://www.easa-award.net/, http://www.bth.se/llab/easa_2002.nsf).

1 Introduction

In numerous areas within computer science, engineering, and operations research, combinatorial optimization problems can be identified whose resolution are of high practical importance. Examples are the traveling salesman problem, knapsack problems, and scheduling and routing problems, to name a few. While the description and explanation of these problems is comparably easy, their resolution is

Please use the following format when citing this chapter:

Geiger, M.J., 2006, in International Federation for Information Processing, Volume 210, Education for the 21st Century-Impact of ICT and Digital Resources, eds. D. Kumar, and Turner J., (Boston: Springer), pp. 65–74.

not, and in many cases is even NP-hard [7]. Algorithms that have been developed for these problems include heuristics and more recently metaheuristics such as evolutionary algorithms, tabu search, and simulated annealing. These techniques solve an underlying problem through successive modification and improvement of alternatives until a solution is identified which cannot longer be improved.

When teaching the principles of modern heuristics, demonstration tools are particularly useful as they allow a monitoring of the intensive computations performed by the algorithm. In addition to theoretical explanations, the progress of the metaheuristics when solving a problem at hand becomes more visual. For students having to face the difficulty of understanding both the problem *and* the search algorithm, this is of great value in order to quickly progress in this complex field of science.

Numerous implementations of algorithms for combinatorial optimization problems have been made available on the World Wide Web. Prominent examples are:

1. The remote interactive optimization testbed RIOT [10].
2. The GA archives [13].
3. The more specialized EMOO webpage for multi-objective optimization problems [4].

The first example maintains a collection of software dedicated to the demonstration of algorithmic approaches for a variety of problems. It consists of web-based applications that visualize the described problems and allow a basic interaction of the user with the system. While the structure of the problems and the general ideas of the algorithm become easily transparent to the user, a further adaptation of the implemented methods is not possible as this is clearly not the idea of the testbed.

The other two examples of software collections bring together research oriented software packages that may be reused by fellow researchers. They comprise highly specialized as well as more generic programming code in a variety of programming languages. In order to be reused, a thorough understanding of the implemented techniques is necessary, and an adaptation of the techniques to particular problems requires in many cases a close examination of the source code. While this does not present a problem to rather experienced researchers, it limits its use in higher education to some extent.

Bringing together research and teaching is especially crucial in this context, as knowledge progresses at a fast pace. In an ideal setting, a system would be available that allows the user not only to study predefined algorithms but also to parameterize own settings and test them on individual problem instances. The current article describes the proposition and implementation of such a system. It is organized as follows. In the following Section 2, the general concepts of local search heuristics are reviewed. A system demonstrating the application of modern metaheuristics to scheduling problems is presented in Section 3, and conclusions and discussion are given in Section 4.

2 Concepts of modern metaheuristics

As sketched above, metaheuristics aim to solve optimization problems through successive modification/improvement of alternatives. While the general principles of the metaheuristics are rather simple, complex interactions of the parameter settings conclude from a variety of operators used within the search algorithms. The main concepts are here illustrated using the example of a flow shop scheduling problem with the data given in Table 1.

Table 1. Example of a flow shop scheduling problem

Job	Operations	Precedence constraints	Processing times
J_1	$\{O_{11}, O_{12}\}$	$O_{11}\ \pi\ O_{12}$	$p_{11}=3, p_{12}=2$
J_2	$\{O_{21}, O_{22}\}$	$O_{21}\ \pi\ O_{22}$	$p_{21}=4, p_{22}=1$
J_3	$\{O_{31}, O_{32}\}$	$O_{31}\ \pi\ O_{32}$	$p_{31}=2, p_{32}=3$
J_4	$\{O_{41}, O_{42}\}$	$O_{41}\ \pi\ O_{42}$	$p_{41}=1, p_{42}=4$

The problem consists of four jobs J_1, ..., J_4, each of them comprising two operations $J_j=\{O_{j1}, O_{j2}\}$ with given processing times p_{jk}. It is assumed that all operations O_{jk} have to processed on machine M_k. The objective of the scheduling problem is to find a schedule, assigning starting times S_{jk} to each operation O_{jk} such that a single or a set of objectives is minimized while all side constraints of the problem, such as the precedence constraints of the operations, are respected. A prominent example of an optimality criterion in this context is the maximum completion time (makespan) $C_{max}=\max(S_{j2} + p_{j2})$.

A closer examination of the problem structure reveals that a schedule may be represented as a sequence of jobs, which on the other hand can be transformed into a schedule by assuming earliest possible execution of the operations with respect to the given job sequence [6]. Figure 1 gives an example of the job sequence $1 \rightarrow 2 \rightarrow 3 \rightarrow 4$ for the problem instance of Table 1. The Gantt chart [16] of the schedule is produced by decoding the particular sequence.

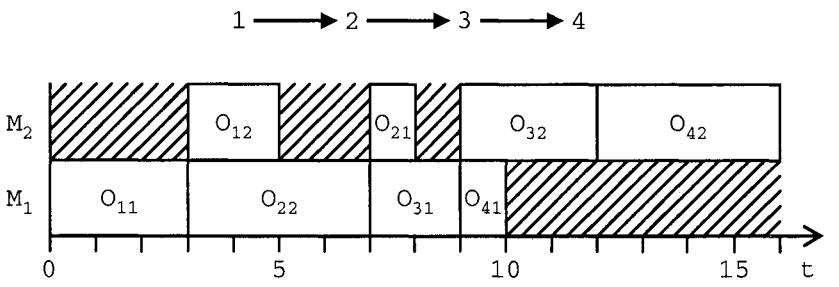

Figure 1. Representation (permutation) and corresponding alternative (schedule)

Metaheuristics work with representations of alternatives as described above, on which operators are executed that perform modification steps to generate new solutions. For the example of representing an alternative as a permutation, operators are used that modify this permutation in a particular way:

- Simple local search operators/mutation operators [12].
  - Forward shift operators, removing a job from position j of the permutation and reinserting it at position k, $k > j$.
  - Backward shift operators, removing a job from position j of the permutation and reinserting it at position k, $k < j$.
  - Exchange operators, exchanging the position of two jobs.
- Crossover operators, recombining the information provided by two permutations and returning two new alternatives on that basis [15].

Due to the limited availability of memory, some of the alternatives generated during search have to be discarded, leading to the necessity of a selection step in the process. Most metaheuristics consider in each step of the search either a single alternative or a set of fixed cardinality, to which sometimes is referred to as a *population*. Especially in the case of multi-objective optimization problems the heuristics need to be able to maintain a set of solutions as due to often conflicting objectives not a single optimal alternative exists but rather a set of Pareto optimal ones [5]. Archiving strategies of identified best solutions play here a particular role and add to the complexity of the search algorithms.

It can be already seen from the brief explanations above, that despite their simplicity, metaheuristics require an extensive setting of parameters. The correct choice of operators is crucial for the performance of the search algorithm, and depends on the problem as such as well as on the particular instance. Experimental investigations have been performed on almost any optimization problem, and approximate recommendations of how to configure a particular metaheuristic are available in the literature. In higher education however, an interactive demonstration of the resolution behavior is still useful as it allows the students to gather hands-on experiences with the algorithms. How such a system has been made available will be the content of the following section.

3 A learning and decision support system for scheduling

3.1 Components

A learning and decision support system for scheduling has been implemented, allowing the resolution of machine scheduling problems under multiple objectives. Its main components are given in Figure 2.

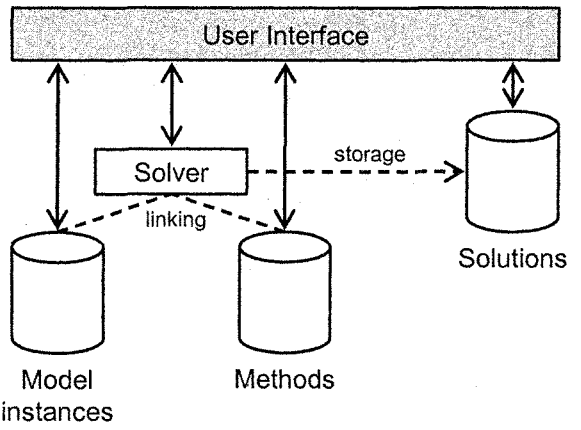

Figure 2. System architecture

The system already includes a database of scheduling benchmark instances taken from the literature [2, 3] while individual data sets may be created by the user, too.

A range of implemented metaheuristics allows the resolution of the scheduling problems. Model instances are linked to methods via a solver which also keeps track of the obtained results. Implemented methods include:

1. Priority rules [9], based on the early work of GIFFLER and THOMPSON [8] for generating active schedules.
2. Local search neighborhoods [12] within a multi-point hillclimber.
3. Multi-objective evolutionary algorithms [1], incorporating elitist strategies and a variety of crossover neighborhoods such as uniform order based crossover, order based crossover, two point order crossover, and partially mapped crossover.
4. The multi-objective simulated annealing algorithm MOSA of TEGHEM et al [14].
5. A module based on the aspiration interactive method AIM [11] for an interactive search in the obtained results.

The whole functionality is made available to the user through a graphical user interface.

3.2 Visual user interface

Both the interaction of the user with the models and algorithms, as well as with the obtained results is possible through a multilingual user interface. Figure 3 shows, on the left, the window allowing the definition of the problem data and on the right the window giving access to the functionalities of the metaheuristics. New configurations of search algorithms may be derived here and tested on the benchmark instances. Necessary parameter settings of the search algorithms can easily be parameterized by selecting the corresponding objects in the window and changing their attribute values.

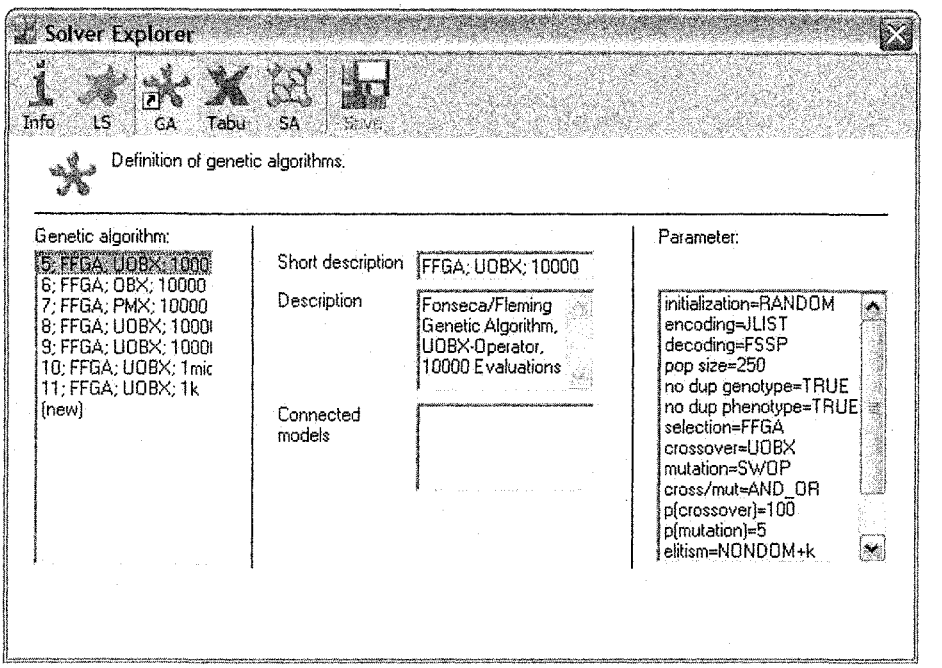

Figure 3. User interface allowing the definition of search algorithms

By providing a graphical user interface for the configuration of the underlying search algorithms, no source code needs to be written nor adapted or recompiled.

3.3 Problem resolution

After the definition of the parameters of the metaheuristics along with the model data, a solver executes the search algorithm on a problem instance and produces results that are stored in a database. It is possible to store split results obtained during search in order to monitor and analyze the progress of the metaheuristic depending on the amount of computations.

The visualization of the results is supported both in outcome and in alternative space:

1. A two-dimensional plot in objective space gives the outcomes of the best found alternatives. On the vertical and on the horizontal axis, one objective of the problem may be plotted at once. An example of such a visualization is given in Figure 4 on the left. Each schedule appears as an object in the outcome plot, with a position depending on its objective function values. The user is enabled to navigate through the plot by either selecting an alternative with the mouse pointer or following an interactive procedure based on the aspiration interactive method AIM [11]. In this procedure, aspiration levels are introduced for each objective function that narrow down the set of alternatives. Only solutions that fulfill the

aspiration levels given by the decision maker are considered to be of interest while the others are successively removed from the decision making procedure.

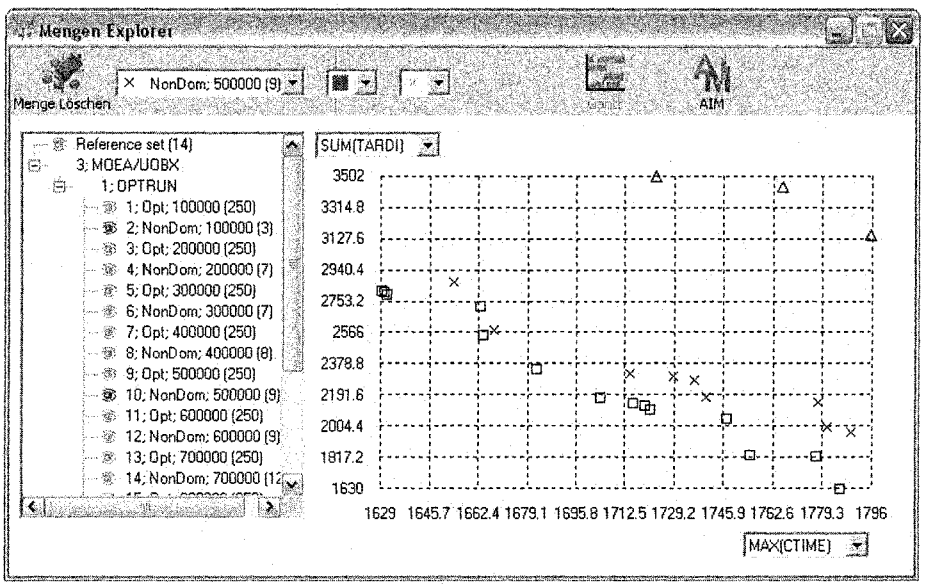

Figure 4. Monitoring the outcomes of a search run after 100,000, 500,000, and 1,000,000 evaluated alternatives. The different stages of the approximation are visualized in outcome space by means of different symbols

2. The second visualization gives a Gantt chart of a selected alternative. Figure 5 gives an example of a schedule. Here, the visualization is job-oriented, plotting the jobs and their corresponding operations in rows, but the orientation may also be changed to a machine-oriented chart. A detailed monitoring of the starting times of all operations is possible, and graphical indications of tardy jobs are given.

The outcome plot may also be used to analyze the progress of the metaheuristics and compare the results of different search algorithms. Figure 4 gives for a test run of an evolutionary algorithm the obtained approximations after 100,000, 500,000, and 1,000,000 evaluations. Each approximation is visualized using some symbol (e.g., $\times$) which may be changed by the user.

It can be seen, that the front of the approximation gets closer to the true Pareto front with increasing number of evaluations. Also, the number of identified alternatives and their coverage of the Pareto front increases, giving the decision maker more alternatives from which to choose from.

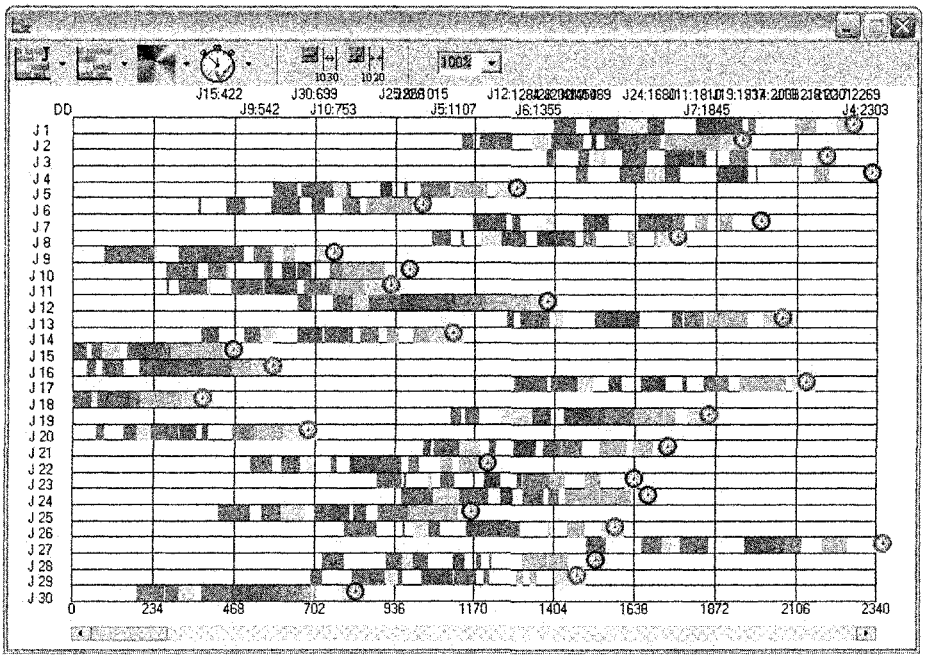

Figure 5. User interface visualizing the Gantt chart of a selected alternative

3.4 Educational use

When teaching modern heuristics such as evolutionary algorithms, traditional educational methods are of limited applicability. While textbooks describing the approaches are available, a thorough understanding of the methods by means of theoretical explanations only, e.g. by stating the corresponding pseudo-codes, is difficult.

Many metaheuristic approaches require an extensive setting of control parameters, and an important resulting aspect is the fact that their behavior can be best studied by experimental tests during which different parameter settings are systematically tested. Consequently, demonstration software is needed that allows such investigations. Ideally, an available system is equipped with a graphical user interface for the direct manipulation of the underlying resolution approaches.

By implementing such a system for the problem domain of scheduling, we enable teachers and students to interactively explore the capabilities of different metaheuristics for the resolution of corresponding problem instances. The understanding of the techniques is further supported by the visualization of the results. Contrary to other class libraries, the output of the results is possible in a visual way, visualizing the Gantt charts of schedules (see Figure 5) and plotting the outcomes of the optimality criteria in an outcome plot (see Figure 4).

In addition to the visualizations of the results, teachers and student may easily modify the data of the investigated problem instances. This includes the data as

such, e.g. the processing times of the operations, as well as the set of optimality criteria.

4 Conclusions and discussion

A system for the resolution of scheduling problems under multiple objectives has been presented. It allows the definition of problem instances and modern heuristics on the basis of an implemented library. Adaptations are possible by setting parameter values. Therefore, no source code needs to be recompiled or changed in order to make the system work. The results may easily be visualized and compared to each other using different plots in alternative and outcome space.

As the system is aimed at end users in higher education and research, all interactions of the user with the system are supported by a graphical interface. So far, seven languages are available for the items of the user interface, namely English, French, German, Hungarian, Italian, Polish, and Spanish. The presented system is freely available for educational and research oriented use. It is accompanied by a 103-page printed manual, and a first chapter containing a tutorial of how to use the system quickly introduces the prospective user into its functionalities.

The presented system is a useful tool for demonstrating the capabilities of metaheuristics for the resolution of scheduling problems under multiple objectives. The flexible architecture within the problem domain makes it usable for a wide range of problem instances with different characteristics. On the other hand however, it is bound to the problem domain of scheduling, and adaptations to other problems are not permitted to the end user. We believe however, that this disadvantage is outweighed by the provided functionalities in the particular problem domain.

The software successfully competed in the finals of the *European Academic Software Award*, held in Ronneby (Sweden). It has been evaluated by an international panel of experts and honored with an award.

Acknowledgements

The author thanks ZSÍROS ÁKOS (University of Szeged, Hungary), PEDRO CAICEDO, LUCA DI GASPERO (University of Udine, Italy), and SZYMON WILK (Poznan University of Technology, Poland) for multilingual versions of the software.

References

1. Tapan P. Bagchi. *Multiobjective scheduling by genetic algorithms*. Kluwer Academic Publishers, Boston, Dordrecht, London, 1999.

2. Matthieu Basseur, Franck Seynhaeve, and El-ghazali Talbi. Design of multi-objective evolutionary algorithms: Application to the flow-shop scheduling problem. In *Congress*

on Evolutionary Computation (CEC'2002), volume 2, pages 1151–1156, Piscataway, NJ, May 2002. IEEE Service Center.

3. J. E. Beasley. OR-library: Distributing test problems by electronic mail. *Journal of the Operational Research Society*, 41(11):1069–1072, 1990.

4. Carlos A. Coello Coello. List of software on evolutionary multiobjective optimization. http://www.lania.mx/~ccoello/EMOO/EMOOsoftware.html.

5. Carlos A. Coello Coello, David A. Van Veldhuizen, and Gary B. Lamont. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Kluwer Academic Publishers, 2002.

6. Richard L. Daniels and Joseph B. Mazzola. A tabu-search heuristic for the flexible-resource flow shop scheduling problem. *Annals of Operations Research*, 41:207–230, 1993.

7. M. R. Garey and D. S. Johnson. *Computers and Intractability – A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, San Francisco, CA, 1979.

8. B. Giffler and G. L. Thompson. Algorithms for solving production-scheduling problems. *Operations Research*, 8:487–503, 1960.

9. R. Haupt. A survey of priority rule-based scheduling. *Operations Research Spektrum*, 11(1):3–16, 1989.

10. Dorit Hochbaum. The remote interactive optimization testbed RIOT. http://riot.ieor.berkeley.edu/riot/.

11. V. Lotfi, T. J. Stewart, and S. Zionts. An aspiration-level interactive model for multiple criteria decision making. *Computers & Operations Research*, 19(7):671–681, 1992.

12. Colin R. Reeves. Landscapes, operators and heuristic search. *Annals of Operations Research*, 86:473–490, 1999.

13. Alan C. Schultz. The GA archives. http://www.aic.nrl.navy.mil/galist/src/.

14. E. L. Ulungu, J. Teghem, P. H. Fortemps, and D. Tuyttens. MOSA method: A tool for solving multiobjective combinatorial optimization problems. *Journal of Multi-Criteria Decision Making*, 8:221–236, 1999.

15. Darrell Whitley. Permutations. In Thomas Bäck, David B. Fogel, and Zbigniew Michalewicz, editors, *Handbook of Evolutionary Computation*, chapter C3.3.3, pages C3.3:14–C3.3.20. Institute of Physics Publishing, Bristol, 1997.

16. James M. Wilson. Gantt charts: A centenary appreciation. *European Journal of Operational Research*, 149:430–437, 2003.