Towards the Benchmarks for Scheduling Problems

Sanja Petrovic

Automated Scheduling, Optimisation and Planning (ASAP) Research Group
School of Computer Science, University of Nottingham
Jubilee Campus, Nottingham NG8 1BB, UK

Introduction

The aim of this paper is not to offer any definite proposal for the scheduling competition, but to discuss some of the key issues to be considered in such an endeavour. The discussion will refer to three types of scheduling problems: production scheduling, employee scheduling (the discussion will be focused on nurse rostering being a very complex and highly constrained scheduling problem) and university timetabling. The reasons for choosing these types of scheduling problems are twofold: they are widely studied in the scheduling community and have led to a wealth of the scheduling models and algorithms, and the author has experience in solving these problems. However, the competition could include other types of scheduling problems, such as project scheduling, etc. First, a brief description of each of these types of problems will be given, together with available benchmark problems. After that, the following issues will be addressed: (a) is there a need for a common formal representation of different scheduling problems, (b) how to perform the evaluation of the scheduling algorithms, (c) is there a need for both randomly generated problem instances and real-world ones, and (d) what conclusions could we draw after the competition.

The paper will also draw upon the experiences gained in the International Timetabling Competition organized by the Metaheuristic Network and sponsored by Patat (a series of conferences on Practice and Theory of Automated Timetabling) that was held in 2003. More information about this competition can be found on the Web page http://www.idsia.ch/Files/ttcomp2002/. The 2nd International Timetabling Competition will be organised in the near future.

Types of Scheduling Problems

Three types of scheduling problems will be considered: production scheduling, employee scheduling, and university timetabling.

A production scheduling problem concerns the allocation of resources (e.g., equipment, materials, labour) to tasks over time under some constraints (Pinedo, 2002). A variety of machine configurations were studied, including single machine, parallel machines, flow shops, job shops, etc. Different criteria such as makespan (i.e. the completion time of the last job), the number of tardy (early) jobs, the average tardiness (earliness) of jobs, and the total flow-time have been typically considered as performance measures.

The employee scheduling is defined as a problem of placing resources into slots in a pattern, subject to given constraints, where the pattern denotes a set of legal shifts defined in terms of work to be done (Wren, 1996). Nurse rostering is a particularly complex and constrained type of employee scheduling problem, which takes into consideration different, usually conflicting constraints on staff qualifications, and a large number of legal, management and staff requirements (Burke et al. 2004).

University timetabling problems can be classified into two main categories: course and examination timetabling. In course (examination) timetabling problems a number of courses (exams) are allocated into a number of available classrooms and a number of timeslots, subject to given constraints (Burke and Petrovic, 2002).

One of the main differences between these three types of scheduling problems is that nurse rostering and university timetabling are in general treated as static problems in the sense that the size of the problem is not changing over time, while production scheduling problems can be both static and dynamic. In the dynamic scheduling problems the size of the problem changes dynamically; for example, the number of machines can change due to unavailability of a machine, new jobs enter the shop floor over time, etc. The further discussion will focus on the static problems.

The literature on scheduling models and algorithms is very rich indeed. In general, the algorithms can be classified into two broad classes: constructive and improvement algorithms (Pinedo, 2002). The constructive algorithms start with an empty schedule and gradually construct a schedule by adding one element (for example, an exam/course, a job, etc) into a schedule at each step. The improvement algorithms start from a complete schedule and then try to improve it with respect to the objective function by manipulating the elements of the schedule.

For each of these three types of problems there exist a number of benchmark problems that are downloadable from the Internet.

There is a number of production scheduling benchmark problems. Two of them that are of particular interest are as follows:

(a) Generated by Demirkol et al., presented in (Demirkol et al. 1996) and downloadable from the Web page http://cobweb.ecn.purdue.edu/~uzsoy/ResearchGroup/Index.html. It contains a wide variety of scheduling problems including single and parallel machine environments, job shop with different routing configurations, problems with dynamic arrivals of jobs, etc. In total, there are almost 10,000 data sets.

(b) Generated by Taillard, presented in (Taillard, 1993) and downloadable from the Web page http://people.brunel.ac.uk/~mastjjb/jeb/orlib/jobshopinfo.html. It contains 260 randomly generated, rather simple scheduling problems. However, the size of these problem instances correspond to real dimensions of industrial problems (starting from 20 jobs 5 machines up to 500 jobs 20 machines).

For a long time, there were no benchmark data sets available for nurse rostering. Recently, an excellent Web page was formed that contained benchmark data sets based on real world data, see http://www.cs.nott.ac.uk/~tec/NRP/. It currently has 13 data sets.

The university timetabling community very often uses a set of benchmark data sets for examination timetabling. They are collected by a number of researchers and can be downloaded from the Web page http://www.cs.nott.ac.uk/~yxb/TTdata/. They are based on real world data but use mostly rather simplified set of constraints and a simple objective function to evaluate the quality of the schedule.

Common Representation of Different Scheduling Problems

The first question that arises in the context of different types of scheduling problems mentioned in Section 1 is what do they have in common? They are characterised by the existence of two sets of constraints, usually referred to as *hard and soft constraints.* Hard constraints must not be violated and they determine the feasibility of solutions. Very often, in practice, it is not possible to generate a solution that satisfies all the soft constraints (also called preference constraints). The extent of violation of soft constraints determines the quality of solutions.

In these three types of scheduling problems, constraints differ very much in their nature. Typical constraints are as follows:

1. Production scheduling: Precedence constraints that are imposed on jobs which define their order of processing on the machines; release and due dates which define the time when a job can start and should finish its processing, respectively, machine availability, etc.

2. Nurse rostering: Cover constraints which determine the minimum number of nurses that must be assigned to a particular shift, the maximum and minimum number of days that a nurse may work in a row, the number of weekends that a nurse may work over a period of time, the preferred working shifts expressed by nurses, etc.

3. University timetabling: There are some common constraints imposed on both university examination and course timetabling, such as: no person can be allocated to be in more than one place at a time, and the total resources required in each time period must not be greater than the resources that are available in that period. However, there is a large number of constraints that differ between the university examination and course timetabling. Typical soft constraints set in the examination timetabling problems are: the exams of each student should be equally spread over the examination period, precedence constraints are imposed on some exams, etc. Typical soft constraints set in the course timetable problems are: students should not have a single class on a day, students should not have more than two classes consecutively, etc.

Considering different types of scheduling problems a question arises whether it would make sense to include all of them in the scheduling competition. There are pros and cons of having different types of scheduling problems in benchmark datasets. Over decades, the scheduling community has been focused on developing algorithms that work well on a particular problem. Very often, they use some domain knowledge in order to generate a high quality solution. Therefore, it is very likely that those types of algorithms could not run on different types of scheduling problems. However, recent years have seen an increase in research towards generating algorithms that work well over a range of scheduling problems, by choosing an appropriate heuristic(s) for a given problem. Such algorithms are usually referred to as "hyper-heuristics" (Burke et al. 2003). Should the competition be open to both types of algorithms? In that case, we could rank separately algorithms for each type of scheduling problems, and also rank algorithms that are developed for solving all different types of scheduling problems.

In order to consider different types of scheduling problems together, there must exist a unique formalism for their representation that will be able to capture a variety of existing domain-specific constraints. In the optimisation/scheduling community there have been some attempts to develop a language for constraint representation, but they addressed only a single type of problem. For example, Kingston (2001), and Reis and Oliveira (2001) designed languages for the formal representation of the constraints in university timetabling problems. Le Pape (1994) presented the software Schedule that was developed by ILOG, which implements a constraint language that is powerful enough to represent a variety of resource and temporal constraints. However, such a representation is more appropriate for a certain type of algorithms, such as constraint programming, which can work directly on this representation. Some other type of algorithms, such as meta-heuristics, may need to map the given representation of the problem into a representation suitable for their running before starting to construct a schedule. Therefore, the time required for mapping should

not be taken into account when measuring the CPU time of algorithm running.

Evaluation of Algorithms

In general, the comparison of algorithms is a multi criteria problem. There are many criteria that have to be taken into account in the evaluation of an algorithm, such as the algorithmic power of the algorithm (i.e. its efficiency and effectiveness), the flexibility and extensibility (for example, how easy is to handle new constraints), learning capabilities (for example, whether the algorithm learns during the search of the solution space and consequently can choose an appropriate heuristic, or whether the algorithm learns from solving previous problems), etc. However, it would be difficult to rank the algorithms with respect to all of these criteria. It is well known that different multi criteria methods can offer different ranking result. Therefore, in order to have a unique rank of algorithms, the evaluation should be restricted to a single criterion. The two straightforward criteria that are of high importance and that are relatively easy to measure are the quality of generated solution(s) and the computation requirements. Although, in some scheduling environments, such as university examination and course timetabling, time does not play an important role because a timetable does not need to be generated quickly, in order to enable a fair comparison the same amount of time (CPU) should be given to all the algorithms. However, participants will run their algorithms on computers of different characteristics and this has to be taken into consideration. In the course timetabling competition mention in Introduction, a program was developed that tested roughly how long the participant was allowed to run his/her algorithm on a particular computer. Therefore, each participant would be given a certain CPU time to run his/her algorithm(s), while the quality of the generated solution would determine the "success" of the algorithm.

In order to have a reliable indicator of the quality of the generated solutions the competition organiser should provide software for calculating the value of the objective function. This implies that a common output format for the solutions must be defined that would be used in this calculation.

Apart from checking the quality of the solutions to the generated benchmark problems the submitted algorithms should run on a number of new problem instances unseen by the competitors. This was the practice in the international course timetabling competition in which three more timetabling problems were defined and used for evaluating the performance of the algorithms. The rationale behind this is to check whether the algorithm is designed to cope well just with the given problem instances or can successfully solve new ones.

Randomly Generated Problems Versus Real-world Problems

The competition should consider both randomly generated benchmark problems and real-world problems. Randomly generated problems enable us to control the properties of the problems. The size of a problem is a general property of all the types of scheduling problems. For example, we can define small, medium and large problems with respect to the number of events to be scheduled in an university timetabling problem (this classification was used in the course timetabling competition). In addition, some more "sophisticated" properties determine the difficulty of a problem. For example, some research work has been carried out into the measuring of difficulty of production scheduling problems (Mattfield et al., 1999, Watson et al., 2003, Streeter and Smith, 2006). This research work could serve as useful guidelines on what properties to consider when generating benchmarks problems. Therefore, randomly generated problems would enable us to test whether an algorithm works well on a wide variety of problems of different properties.

On the other hand, solving real-world problems is (often) our ultimate goal. However, there are some inherent problems with using real-world benchmarks. First, they are often confidential and industrial collaborators do not want to make them publicly available for a variety of reasons. Second, real-world problems are often very complex and require a synergy of algorithms. For example, if the shop floor is overloaded a decision can be taken to include a night shift to complete jobs that have to be delivered urgently, which is an extra task on top of the scheduling decisions. Third, underlying almost every real-world scheduling problems are activities fraught with uncertainties (for example, a machine can breakdown, a job can take longer time to process, a required material for job processing is not available, etc). In such uncertain environments, rescheduling plays an important role. However, rescheduling raises a variety of additional research issues and in author's opinion it should be left out of this competition. In spite of all the problems with real-world data they are certainly of high value for the competition. Some real-world problems have structures that are difficult to obtain by randomly generated data, and consequently algorithms may exhibit different performance on them.

Conclusions after the Competition

This competition should serve as an excellent test bed for a wide range of algorithms for scheduling that have been developed or are under development. However, no definite conclusion about the superiority of a single algorithm should be drawn. No matter how carefully designed, our benchmark problems cannot cover a wide variety of scheduling problems that exist in the real-world. Although, one algorithm can have excellent performance on our

benchmark problems, for some other benchmark problems with properties different from the ones that we considered, some other algorithms might be more suitable. Ideally, the algorithm developers should use the benchmark data sets to analyse which type of problems their algorithms can handle well and to compare their results with the results obtained by using other algorithms.

References

Burke, E. K., De Causmaecker, P., Vanden Berghe, G. and Van Landeghem, H. 2004. The state of the art of nurse rostering, *Journal of Scheduling* 7(6): 441–499.

Burke, E., and Erben, M. eds. 2001. *Lecture Notes in Computer Science*, Vol. 2079: Springer.

Burke, E.K., Kendall, G., Newall, J., Hart, E., Ross P., and Schulenburg S. 2003. Hyper-heuristics: An Emerging Direction in Modern Search Technology, in Glover F., and Kochenberger G. (eds) *Handbook of Meta-heuristics***,** Chapter 16, 457-474: Kluwer.

Burke, E.K., and Petrovic, S. Recent Research Directions in Automated Timetabling, *European Journal of Operational Research - EJOR* 140(2): 266-280.

Demirkol, E., Mehta, S.V., Uzsoy, R. 1998. Benchmarks for Shop Scheduling Problems, *European Journal of Operational Research – EJOR* 109: 137-141.

Le Pape, C. 1994. Implementation of Resource Constraints in ILOG SCHEDULE: A Library for the Development of Constraint-Based Scheduling Systems, *Intelligent Systems Engineering* 3(2): 55-66.

Kingston, J. 2001. Modelling Timetabling Problems with STTL, in (Burke and Erben, 2001), 309-321: Springer.

Mattfeld, D.C. Bierwirth, C., and Kopfer, H. 1999. A search space analysis of the Job Shop Scheduling Problem, *Annals of Operations Research* 86: 441–453.

Pinedo M. 2002. *Scheduling: Theory, Algorithms, and Systems*, Prentice Hall.

Reis L.P., Oliveira,E. 2001. A Language for Specifying Complete Timetabling Problems, in (Burke and Erben, 2001), 322-341: Springer.

Streeter, M. J. and Smith, S. F. 2006. How the Landscape of Random Job Shop Scheduling Instances Depends on the Ratio of Jobs to Machines, *Journal of Artificial Inteligence Research* 26: 247-287

Watson, J-P. Beck, C, Howe, A. Whitley D. 2003. Problem Difficulty for Tabu Search in Job-Shop Scheduling, *Artificial Intelligence* 143(2): 189 – 217.

Wren, A. 1996. Scheduling, timetabling and rostering - a special relationship? In Burke E. and Ross P. (eds), *Lecture Notes in Computer Science* Vol. 1153, 46–75: Springer.

Web Pages:

(International Course Timetabling Competition)
http://www.idsia.ch/Files/ttcomp2002/

(nurse rostering benchmarks)
http://www.cs.nott.ac.uk/~tec/NRP/

(production scheduling benchmarks)
http://cobweb.ecn.purdue.edu/~uzsoy/ResearchGroup/Index.html

http://people.brunel.ac.uk/~mastjjb/jeb/orlib/jobshopinfo.html

(university timetabling benchmarks)
http://www.cs.nott.ac.uk/~yxb/TTdata/