Production Scheduling Problems and Computational Complexity

by George Zobolas, Doctoral Candidate

Production systems are vital for the sustained development and prosperity of modern societies. Production activity constitutes the basis upon which the economic strength and vitality of a country can rely (Sipper and Bulfin, 1997). Although production systems, in a primitive form, appeared as early as 3000bc, the task of creating, operating and maintaining a production system has become increasingly complex after the first half of the 20th century. On the one hand, the culprit for this increased complexity is technology advancements at an exponentially, most of the times, accelerating pace which make production systems obsolete often before their depreciation period ends (Hopp and Spearman, 1996). On the other hand, the increased demand for product diversity, very short lead times, and high quality levels put a strain on the production engine which requires a very competent management for proper operation. Fortunately, advanced information technology, mainly in the form of Enterprise Resource Planning (ERP) systems, facilitates production planning, scheduling and control. ERP systems emerged from legacy MRP and MPRII manufacturing systems and they constitute an integrated enterprise information system that automates the flow of material, information and financial resources among all functions within an enterprise using a common database.

Although modern ERP systems are an excellent platform to monitor and control all enterprise operations, they do not always manage to tackle effectively production scheduling problems. Such problems are usually very complex and require the development of specialized solution methodologies which although they can be implemented in an ERP system, they cannot cope with all possible production environments.

Scheduling Problems and Their Complexity

Generally, production scheduling problems involve the allocation of (limited) resources to specific tasks in such a way that a specific criterion is optimized. These problems can be classified based on resource interconnection, operation sequence, precedence constraints, existence of alternative resources and optimization criterion. The most common scheduling problems are the shop type scheduling problems where a set of jobs must be processed on a set of resources (machines). Based on the sequence of processing for each job, the three main types of shop scheduling problems are:

- **Flow shop** scheduling problem (FSSP). The FSSP consists of a set of n jobs to be processed on a set of *m* machines. Each job consists of *m* operations and all jobs must be processed on every machine in the same sequence (Hejazi and Saghafian, 2005).
- **Open shop** scheduling problem (OSSP). The OSSP consists of a set of *n* jobs to be processed on a set of *m* machines. Each job consists of *m* operations and all jobs must be processed on every machine at any order (Liaw, 2000).
- **Job shop** scheduling problem (JSSP). The JSSP consists of a set of *n* jobs to be processed on a set of *m* machines. Each job consists of *m* operations and all jobs must be processed on every machine at a predetermined order, explicitly given for each job (Jain and Meeran, 1999).

Shop scheduling problems are inherently very complex and normally belong to the NP-Hard class, i.e. no polynomial time algorithm can be found to solve them (Conway, 1967). A seemingly easy job shop problem with ten jobs and ten machines (10x10) detained researchers for over 25 years until it was finally solved in mid 80's (Fisher and Thompson instance FT10). Today, there are instances of slightly larger dimensionality (20x15) which are still unsolved (Jain and Meeran, 1999). To highlight the extent of such problems' complexity, a small 10x10 job shop instance has $(10!)^{10} \approx 4 \times 10^{65}$ possible solutions.

Solution Methods

The solution of a shop scheduling method lies in finding an optimum arrangement of operations that optimizes a given performance criterion (usually the minimization of makespan, i.e. the completion time of the last operation) while at the same time satisfying all capacity and precedence constraints. Every distinct arrangement constitutes a feasible solution and all feasible solutions constitute the solution space. This space has to be explored effectively and efficiently so the best solution can be located. Shop scheduling problems can be solved/optimized using two different approaches. The first approach is by using ***complete*** or ***exact*** algorithms. The second approach is applied to complex problems, leads to near-optimum solutions and is characterized by the use of ***approximate*** algorithms. The second approach can be further divided to ***heuristic*** and ***metaheuristic*** methods.

Exact or ***complete*** algorithms are guaranteed to find an optimal solution in bounded time. Yet for the typical scheduling problems (which as mentioned are usually NP-Hard) no polynomial time algorithms exist and therefore exact algorithms might need exponential and impractically long computational time. The family of exact methods is considerably large but the most common exact/complete method is the branch and bound algorithm. Other exact methods are mixed integer programming and decomposition methods.

On the other hand, in ***approximation*** methods, the guarantee of finding optimal solutions is sacrificed in order to get near-optimum solutions in reasonable and practical computational times. The basic form of approximation algorithms is called '***heuristics***', a name derived from the Greek word '*ευρίσκειν*' which means 'to find'. The usual classification of heuristic methods distinguishes Constructive and Local Search methods (Blum and Roli, 2003). *Constructive* algorithms generate solutions from scratch by adding components to an initially empty partial solution. These are typically the fastest approximate algorithms but the speed advantage is counterbalanced by their relatively low performance in terms of solution quality. Local Search algorithms start from an initial solution (possibly generated randomly or by another heuristic) and iteratively try to replace part or the whole solution with a better one in an appropriately defined neighborhood of the current solution (Tarantilis and Kiranoudis, 2002). The main drawback of basic Local Search is that it gets easily trapped in local optima, i.e. regions in the solution space where the global optimum is not located.

During the last decades, a new kind of approximate algorithm has emerged. This new type of algorithm basically combines heuristic methods in higher level frameworks. The aim of the new methodology is to efficiently and effectively explore the search space driven by logical moves and knowledge of the effect of a move, many times mimicking the way living creatures think and behave (Artificial Intelligence). These methods are nowadays called ***metaheuristics***, a term introduced by Glover in 1986. The most apposite definition though was given by Stutzle in 1999:

> *"Metaheuristics are typically high-level strategies which guide an underlying, more problem specific heuristic, to increase their performance. The main goal is to avoid disadvantages of iterative improvement and, in particular, multiple descents by allowing the local search to escape from local optima. This is achieved by either allowing worsening moves or generating new starting solutions for the local search in a more intelligent way than just providing random initial solutions. Many of the methods can be interpreted as introducing bias such that high quality solutions are produced quickly. This bias can be of various forms and can be cast as descent bias (based on the objective function), memory bias (based on previously made decisions) or experience bias (based on prior performance). Many of the metaheuristic approaches rely on probabilistic decisions made during the search. But, the main difference to pure random search is that in metaheuristic algorithms randomness is not used blindly but in an intelligent, biased form."*

Metaheuristic methods have an advantage over simpler heuristics in terms of solution robustness; however they are more difficult to implement and are problem specific. Among all metaheuristics methodologies proposed, the most widely known and used are:

- *Simulated annealing* (SA) proposed by Kirkpatrick et al. (1983). SA mimics a natural phenomenon, the annealing of hot metals. The fundamental idea is to allow worsening moves during solution space search in an attempt to escape local optima and move to other more promising regions.
- *Genetic algorithm* (GA) proposed by Holland (1992). GA is inspired by nature's capability to evolve living beings well adapted to their environment. The basic idea behind GA is the mating of good solutions to form new solutions of hopefully better performance. Escape from a strictly defined subspace of the global search space is performed by a mutation operator which randomly changes individual attributes.
- *Neural networks* are an advanced artificial intelligence technology that mimics the brain's learning and decision making process (Foo and Takefuji, 1988). A neural network consists of a number of nodes with neuron connections between the nodes and can be used to extract patterns and detect trends that are too complex to be noticed by either humans or other computer techniques.
- *Tabu search* proposed by Glover (1989) explicitly uses the history of the search, both to escape from local optima and to implement an explorative strategy. Basic tabu search uses a short-term memory to store all recent moves to escape from local optima and avoid circles around it.

A rapidly growing research trend in the field of combinatorial optimization is the adaptation of ***hybrid*** metaheuristic approaches, which combine different concepts or components of various metaheuristics (Blum and Roli, 2003). Hybridization, when correctly applied, may further enhance the effectiveness of the solution space search and may overcome any inherent limitations of the single metaheuristic algorithms used. Therefore new opportunities emerge which may lead to even more powerful and flexible search methods.

Conclusion and Areas of Further Research

In conclusion, production scheduling has traditionally been an area of great academic and industrial interest. Technological advancements in information technology have resulted in the development of advanced IT frameworks to deal with complex scheduling problems in a struggle to minimize production costs and lead times. Although scheduling problems are considered very hard to solve, modern metaheuristic methods (single or hybrid) constitute a solid, effective and efficient solution framework. Of course, the development of even

more powerful methods is of great importance for contemporary production systems and additional research is essential towards the development of methods adaptable to a wide range of scheduling problems. The increased flexibility of such methods would in turn facilitate their integration to modern ERP packages providing the necessary link between academic research and industrial practice.

REFERENCES

- Blum, C. and Roli, A. (2003) Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison, *ACM Computing Surveys*, v35, 268-308
- Conway, R.W., Maxwell, W.L., Miller, L.W. (1967) *Theory of scheduling*, Addison-Wesley
- Foo, S.Y. and Takefuji, Y. (1988) Neural Networks for solving job-shop scheduling: Part1. Problem representation. *Proceedings of IEEE IJCNN*, v2, 275-282
- Glover, F. and Greenberg, H.J. (1989) New approaches for heuristic search: A bilateral linkage with artificial intelligence. *European journal of operational research*, v39, 119-130
- Hejazi, S.R., Saghafian, S., (2005) Flowshop-scheduling problems with makespan criterion: a review, *International Journal of Production Research*, v43, 2895-2929
- Holland J.H (1992) *Genetic algorithms*, Science Am
- Hopp, W.J. and Spearman, M. L. (1996) *Factory Physics: Foundations of Manufacturing Management*, Irwin
- Jain, A.S. and Meeran, S. (1999) Deterministic job-shop scheduling: Past, present and future, *European Journal of Operations Research*, v113, 390-434
- Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P., (1983) Optimization by simulated annealing. *Science*, 4598, 671-680
- Liaw, C.-F. (2000), *A hybrid genetic algorithm for the open shop scheduling problem*, European Journal of Operational Research, v124, 28-42.
- Sipper, D. and Bulfin, R. L. (1997) *Production Planning, Control and Integration*, McGraw Hill
- Tarantilis, C.D. and Kiranoudis, C.T. (2002) A list-based threshold accepting method for the job-shop scheduling problems, *International journal of production economics*, v77, 159-171