Comparison of multi-objective approaches to the real-world production scheduling

Gregor Papa*
Computer Systems Department, Jožef Stefan Institute
Jamova c. 39, SI-1000 Ljubljana, Slovenia
Email: gregor.papa@ijs.si

Peter Korošec
Computer Systems Department, Jožef Stefan Institute
Jamova c. 39, SI-1000 Ljubljana, Slovenia
Email: peter.korosec@ijs.si

Summary

The multi-objective optimization approach has a large influence in the industrial production scheduling. The goal of such optimization is to find a production schedule that satisfies different, usually contradictory, production and business constraints. We implemented a memetic version of several multi-objective algorithms with different approaches to problem solving. The customized reproduction operators and local search procedures were also used. These memetic algorithms were applied to real order-lists from a production company. We show that the multi-objective approaches are able to find high-quality solutions, also when quick respond is required to adapt to dynamic business conditions.

Keywords: *combinatorial optimization, job shop scheduling, multicriteria, memetic, genetic algorithm.*

1 Introduction

In the past we have already successfully approached a production-scheduling problem with a single-objective optimization.[1] The optimization goal was to find a production schedule that satisfies the production time constraints and minimizes the production costs. This involved many specific constraints that had to be considered. Later, the problem evolved, which brought some new constraints and new deciding criteria. Since the single-objective approach proved to be inefficient, we had to consider a multi-objective approach.[2]

There was some initial investigation performed on the usage of multi-objective approaches. In the previous work[2] we used the Indicator-Based Evolutionary Algorithm (IBEA),[3] since it nicely upgrades on our initial work when solving the single-objective scheduling problem.[1] In current work we further improve the findings presented previously[2] with the comparison of Non-dominated Sorting Genetic Algorithm-II (NSGA-II),[4] and Strength Pareto Evolutionary Algorithm 2 (SPEA2),[5] and IBEA. Following the IBEA's proven performance for more than three objectives,[6] and the findings that for four contradictory objectives many classic multi-objective approaches are inappropriate,[7] we decided to check the performance of those three multi-objective algorithms with the real-world production problem.

2 Related work

Due to the growing complexity of the real-world scheduling problems, significant work has been devoted to the automation of scheduling and planning processes, where we often have to deal with very large search spaces, real-time performance demands, and dynamic environments. Effective production scheduling can result in reduction of manpower and production costs by minimizing machine idle time and increasing the number of on-time job deliveries.

Multi-objective optimization[8] is very common within the world of engineering problems. These problems with multiple objectives are also recognized in solving of planning and scheduling problems.

Memetic Algorithms (MAs) were developed to obtain even better results than the genetic algorithm (GA) for various scheduling applications, and with the use of local search techniques the results were further improved. Such an approach not only improves the quality of the solutions, it also reduces the overall computational time.[9]

In one of our initial work[1] a guided local search algorithm was tested on real-world test cases of a production-scheduling problem. Such a problem is a member of the family of job shop scheduling problems,

which are known to be NP-hard. Due to the problem's complexity (many constraints) specialized local searches were used. They were guided with the genetic algorithm, parameter-less evolutionary search, and random selection. It was shown that the use of stochastic approaches greatly improved the quality of the production schedules with respect to the expert's manual solution. Furthermore, the evolutionary approach proved to be notably superior to the random search approach. On the other hand, the random-guided, local search approach was able to come impressively close to the results of the evolutionary approaches. It was obvious that its success was due to the quality of the local searches. Namely, to get good results in a relatively short time, a very powerful set of local searches had to be implemented. This led to good performances for all the guided approaches; the genetic algorithm being the most stable while producing the best results.

In the previous work[2] we have shown that the use of the memetic, multi-objective approach, based on the IBEA, does not reduce the quality of any objective with regard to the lexicographic evaluation of a single-objective approach, when used on the same production-scheduling problem. The only major downside of such an approach is in the increased time that is needed for a good Pareto front of solutions to be constructed.

3 Implemented Multi-Objective Algorithms

Based on the evolved production-schedule requirements we implemented and tested three memetic implementations based on different multi-objective algorithms: NSGA-II, SPEA2, and IBEA. We adapted the basic implementation of these algorithms with our implementations of crossover and mutation operators in order to fully adapt to the specific problem of production scheduling.

3.1 Non-dominated Sorting Genetic Algorithm-II

The NSGA-II is the second version of the famous "Non-dominated Sorting Genetic Algorithm" for solving non-convex and non-smooth single and multi-objective optimization problems. Its main features are: A non-dominated sorting procedure where all individuals are sorted according to the level of non-domination; It implements elitism which stores all non-dominated solutions, and hence enhancing convergence properties; It adapts a suitable automatic mechanics based on the crowding distance in order to guarantee diversity and spread of solutions; Constraints are implemented using a modified definition of dominance without the use of penalty functions.

The NSGA-II orders the population into a hierarchy of non-dominated Pareto fronts. It calculates the crowding distance between members of each front on the front itself. The crossover and mutation are performed as classical operators of the GA. The members of the population are discriminated according the rank of the front and then distance within the front.

3.2 Strength Pareto Evolutionary Algorithm 2

The SPEA2 is one of the most important multi-objective evolutionary algorithms that use elitism approach. Each individual is assigned a raw fitness calculated on the basis of the strength value of solutions who dominate it. To discriminate between individuals having identical raw fitness values additional density information is incorporated. To form the next generation, SPEA2 combines offspring and current population. Subsequently, the best individuals in terms of non-dominance and diversity are chosen. For diversity preservation, SPEA2 uses a truncation procedure.

The SPEA2 calculates the raw fitness as the sum of the strength values of the solutions that dominate a given candidate, where strength is the number of solutions that a given solution dominates. Then the density of an area of the Pareto front is estimated as where is the Euclidean distance of the objective values between a given solution the kth nearest neighbor of the solution, and is the square root of the size of the population and archive combined. It iteratively fills the archive with the remaining candidate solutions in order of fitness. The most similar solutions are truncated form the archive population. For selection of parents some classical GA selection method ,such as binary tournament selection, is used. The crossover and mutation are performed as classical operators of the GA.

3.3 Indicator-Based Evolutionary Algorithm

The IBEA is a multi-objective version of a GA, where the selection process is based on quality indicators. An indicator function assigns each pareto-set approximation a real value that reflects its quality. The optimization goal becomes the identification of a pareto-set approximation that minimizes an indicator function. The main advantage of the indicator concept is that no additional diversity-preservation mechanisms are required. The authors of[3] demonstrated that an indicator-based search can yield results that are superior to popular algorithms such as the improved SPEA2 and NSGA-II.

In a basic version of the IBEA it performs binary tournaments for the selection of individuals to undergo recombination. Next, it iteratively removes the worst individual from the population and updates the fitness values of the remaining individuals.

4 Production-scheduling problem

The scheduling problem was introduced while designing the application for scheduling and monitoring production in the company Eta Cerkno d.o.o. , which produces components for domestic appliances, including hot plates, thermostats and heating elements.[2]

Among the various production stages, the most demanding was the production of cooking hot plates. Here, the fabrication process for components used in different types of plates is similar, but due to clients' demands the models differ in size (height, diameter), connector

type, and power characteristics (wattage). For logistic reasons the clients group different models of plates within the same order, implying the same due-dates for different products. Therefore, their production must be scheduled very carefully to fulfil all the demands (quantities and due-dates), to maintain the specified amounts of different models in stock, to optimally occupy their workers, and to make efficient use of all the production lines. Although the assignment of due-dates is usually performed separately, and before the production scheduling, there are strong interactions between the two tasks. Each order placed by the customer somehow defines a batch of jobs, and their completion times should be as close as possible in order to reduce the waiting time and cost.[10] Furthermore, not all the production lines are equal, since each of them can produce only a few different models. A detailed formulation of the production-scheduling problem is presented in our initial work.[1]

4.1 Production-schedule encoding

The production schedule is encoded into a chromosome with tuples of values, where each tuple (gene) consists of the index of the enumerated order and its assigned production line. A general representation of a chromosome, which includes the encoded production schedule of n orders, is presented in Equation 1.

$$C = g_{11}g_{12} \quad g_{21}g_{22} \quad \cdots \quad g_{n1}g_{n2}, \tag{1}$$

where n is the number of product orders, g_{k1} is an index on some operation $o_k \in O$ and g_{k2} is the production line used to produce the order o_k, for every $k \in \{1,2,\ldots,n\}$.

4.2 Population initialization

The input orders that need to be processed are firstly sorted according to their due-dates into the initial order list O. Next, different chromosomes are constructed as variations of the initial list. Each variation of the indexes of orders from the initial list is encoded as a chromosome. The initial population P consists of N_P chromosomes. In each chromosome the orders are randomly distributed, and also the assigned production line is chosen randomly from among the possible lines for each order.

Since the numbers that are encoded in the chromosome represent the indexes of orders, their values cannot be duplicated and also no number can be missed. The assigned values for the production line always depend on the possible production lines for that order. These conditions must be considered during the initialization and during all the subsequent phases.

4.3 Selection operator

In each iteration mates are selected to undergo the reproduction operators. The selection model is a binary tournament,[11,12] and the selecting criterion is based on the quality indicator.[3] The indicator is used to compare two single solutions. The solution to be selected from the comparison pair is the one with the better indicator value according to the current population.

Similarly, the solutions to be deleted from the merged population of parents and offspring are those that have the worst indicator value according to the current population. Here, the goal is to delete the solution with the smallest degradation of the overall quality of the population.

4.4 Reproduction operators

The order-based Crossover(P, p_c) operator is performed with the interchange of positions that store the ordered numbers within the range. An order-based crossover takes the random part of two parents, swaps the genes of the parents in this part and orders the remaining genes in the first parent in accordance with its order in the second parent. We implemented four types of order-based crossover operators: order (OX), cycle (CX), partially-mapped (PMX) and PTL crossover. During the optimization process they are switched every 10 generations. OX, CX and PMX are more precisely explained in,[11] while PTL is explained in.[13] In OX, PMX, and PTL the two-point crossover scheme is used, where chromosome mates are chosen randomly.

During the Mutation(P, p_m) process each value of the chromosome mutates with a mutation probability p_m. Five different types of mutation are applied:

- changing of the production line - a randomly chosen order is moved from one production line to another one (according to the possible production lines for that order);
- switching of two genes in the chromosome - two randomly chosen orders switch their positions, i.e., one is moved backward (it is produced later), and the other one is moved forward (it is produced earlier);
- shifting of a gene into some new position - a randomly chosen order is moved to some new position. All the orders between the old and new positions of a moved order are shifted. Note that the shifting of a gene into some new position has an effect on a larger part of the chromosome. If an order is moved forward then all the orders between the new and the old position are moved backward, but if an order is moved backward then all the orders between the old and the new position are moved forward.
- replacing similar products - if any consecutive orders of similar products on the production line have descending due-dates (the earlier produced order has a later production due-date than the comparison one) then they are switched.
- merging of similar products - on the randomly chosen production line part of the orders (randomly chosen) are merged according to the different properties of the products (dimensions and power characteristics).

The first mutation type influences the second part of the gene; the second mutation type influences the whole gene; the remaining three mutation types influence only the first part of the gene.

In general, each new population is better than the previous one. To limit a possible disruptive effect of mutation during the later stages of the optimization and to speed up the convergence to the optimum solution in the final optimization stages, the crossover and mutation probability are decreased with each new restart of the algorithm. The smaller number of mutated positions in the later stages can be interpreted as a kind of local search with minor movements around the current solution (i.e., exploitation).

4.5 Fitness evaluation

After the reproduction operators and local search procedures modify the solutions $p \in P$ the solutions are evaluated, so the selection process in the next iteration can be performed. Each solution p defines its set of objective values. A set of n_{obj} objective values is defined with: the number of delayed orders (n_{orders}); the sum of delayed days of all the delayed orders (n_{days}); the required number of workers (n_{workers}); and the sum of the change-over downtime in minutes (t_{change}). The objective values are calculated by the objective functions $f_k, k \in \{1,...,n_{\text{obj}}\}$.

According to,[14] a binary quality indicator can be regarded as a continuous extension of the concept of pareto dominance on sets of objective vectors. The indicator value I quantifies the difference in quality between two objective vectors. In our implementation we used the additive $I_{\varepsilon+}$-indicator[14] (see Eqn. (2)), which gives the minimum distance by which a pareto-set approximation needs to be, or can be, translated in each dimension of the objective space such that another approximation is weakly dominated.

$$I_{\varepsilon+}(p_i,p_j) = min_{\varepsilon}\left\{f_k(p_i) - \varepsilon \leq f_k(p_j) \text{for } k \in \{1,...,n_{\text{obj}}\}\right\} \quad (2)$$

The population P represents a sample of the decision space, and the fitness assignment tries to rank the population members according to their usefulness regarding the optimization goal. Among the different ways of exploiting the information given by P and $I_{\varepsilon+}$, we sum up the indicator values for each population member with respect to the rest of the population. This fitness value F (Eqn (3)) that needs to be maximized is a measure of the loss in quality if a solution p_i is removed from the population.

$$F(p_i) = \sum_{p_j \in P \setminus \{p_i\}} -e^{-I_{\varepsilon+}(p_j,p_i)/\kappa} \quad (3)$$

The parameter κ is a scaling factor and in our case it was set to 0.05, as suggested in.[3]

4.6 Ending condition

In general the algorithm is run until the user stops the optimization process. So the EndingCondition() function checks to see whether a user pressed the stop button. To mimic overnight running we decided to limit the number of evaluations to 300 million instead of checking for the user to press the button.

5 Local search

Local search (LS) procedures are optimization methods that improve the solution by maintaining a currently best one, and exploring the search space step by step within its neighborhood. Usually, the current solution is replaced by a better one in the neighborhood, which is used as a new current solution in the next iteration. This process is repeated until a stopping condition is met, i.e., if there is no better solution within the neighborhood. The interesting point of LS is the fact that it may effectively and quickly explore the basin of attraction of the optimum solutions, finding an optimum with a high degree of accuracy and within a small number of iterations. In fact, these methods are a key component of the metaheuristics that are the state-of-the-art for many optimization problems.

Compared to,[1] where only four LS procedures were used, the approach in the previous[2] and in this paper uses four additional LS procedures to implement the additional requirements and constraints, and also one of the previous procedures was upgraded.

LS is used to improve new solutions on the pareto front and is implemented with eight different procedures, which run sequentially. The reason for such an order of procedures is that the influences of the changes made by, e.g., “stock replacing”, are much more degrading to the current solution than the changes made by “similar product merging”. This means that at the beginning we allow for major changes, which are then more refined by smaller changes. This enables us to find new solutions, which can be located in some other parts of the solution space and still be better than the current solution. The influence of such a LS, due to the very complex nature of our problem, should be a much improved convergence and quality of the obtained solutions compared to the basic optimization technique alone.

6 Memetic Algorithms

There are several approaches to implementing the LS procedures. In our case we merged the presented NSGA-II, SPEA2 and IBEA algorithms to guide the LS procedures. The basic algorithms are implemented with the use of appropriate Java classes of the jMetal framework. Furthermore, since we are dealing with a combinatorial problem, we implemented our problem-specific versions of the crossover and mutation operators. Next, we added the local search procedures to enhance the efficiency of the algorithm.

7 Results

Currently the results are being produced and their interpretation will be available in the final version of the paper.

References

[1] Papa, G., Vukašinović, V., and Korošec, P. Guided restarting local search for production planning. *Engineering Applications of Artificial Intelligence* **25**(2), 242–253 (2012).

[2] Korošec, P., Bole, U., and Papa, G. A multi-objective approach to the application of real-world production scheduling. *Expert Systems with Applications* **40**(15), 5839–5853 (2013).

[3] Zitzler, E. and Künzli, S. Indicator-based selection in multiobjective search. In *Proc. 8th International Conference on Parallel Problem Solving from Nature (PPSN VIII)*, 832–842. Springer, (2004).

[4] Deb, K., Agrawal, S., Pratap, A., and Meyarivan, T. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: Nsga-ii. In Parallel Problem Solving from Nature PPSN VI, Schoenauer, M., Deb, K., Rudolph, G., Yao, X., Lutton, E., Merelo, J., and Schwefel, H.-P., editors, volume 1917 of *Lecture Notes in Computer Science*, 849–858. Springer Berlin / Heidelberg (2000).

[5] Zitzler, E., Laumanns, M., and Thiele, L. SPEA2: Improving the strength pareto evolutionary algorithm for multiobjective optimization. In Evolutionary Methods for Design Optimization and Control with Applications to Industrial Problems, Giannakoglou, K. C., Tsahalis, D. T., Périaux, J., Papailiou, K. D., and Fogarty, T., editors, 95–100 (International Center for Numerical Methods in Engineering, Athens, Greece, 2001).

[6] Wagner, T., Beume, N., and Naujoks, B. Pareto-, aggregation-, and indicator-based methods in many-objective optimization. In Evolutionary Multi-Criterion Optimization, Obayashi, S., Deb, K., Poloni, C., Hiroyasu, T., and Murata, T., editors, volume 4403 of *Lecture Notes in Computer Science*, 742–756. Springer Berlin Heidelberg (2007).

[7] Ishibuchi, H., Tsukamoto, N., and Nojima, Y. Evolutionary many-objective optimization: A short review. In *IEEE Congress on Evolutionary Computation, CEC 2008*, 2424–2431, june (2008).

[8] Deb, K. *Multi-Objective Optimization using Evolutionary Algorithms*. Wiley-Interscience Series in Systems and Optimization. John Wiley & Sons, Chichester, (2001).

[9] Kamrul Hasan, S. M., Sarker, R., Essam, D., and Cornforth, D. Memetic algorithms for solving job-shop scheduling problems. *Memetic Computing* **1**(1), 69–83 (2009).

[10] Zhang, R. and Wu, C. A hybrid local search algorithm for scheduling real-world job shops with batch-wise pending due dates. *Engineering Applications of Artificial Intelligence* **25**(2), 209 – 221 (2012). Special Section: Local Search Algorithms for Real-World Scheduling and Planning.

[11] Michalewicz, Z. and Fogel, D. *How to Solve It: Modern Heuristics*. Springer-Verlag, Berlin / Heidelberg, 2nd edition, (2004).

[12] Bäck, T., Fogel, D., and Michalewicz, Z. *Evolutionary Computation 1: Basic Algorithms and Operators*. Taylor & Francis Group, Heidelberg, 2nd edition, (2000).

[13] Czogalla, J. and Fink, A. On the effectiveness of particle swarm optimization and variable neighborhood descent for the continuous flow-shop scheduling problem. In Metaheuristics for Scheduling in Industrial and Manufacturing Applications, Xhafa, F. and Abraham, A., editors, volume 128 of *Studies in Computational Intelligence*, 61–89. Springer Berlin / Heidelberg (2008).

[14] Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C. M., and Grunert da Fonseca, V. Performance Assessment of Multiobjective Optimizers: An Analysis and Review. *IEEE Transactions on Evolutionary Computation* **7**(2), 117–132 (2003).