HYBRID PARAMETER-LESS EVOLUTIONARY ALGORITHM IN PRODUCTION PLANNING

Vida Vukašinović, Peter Korošec and Gregor Papa
Computer Systems Department, Jožef Stefan Institute, Ljubljana, Slovenia

Keywords: Production, Scheduling, Hybrid algorithm.

Abstract: In the real-world production planning problems there are many constraints that need to be considered. Usually, these constraints and interdependent and the optimization algorithms has to efficiently solve that. This paper presents the hybrid parameter-less evolutionary algorithm used for construction of an optimal production plan. The algorithm is based on genetic algorithm, but is modified to work without the parameter setting. All algorithm control parameters are automatically determined during the optimization. The algorithm was able to solve the constraints and to make an optimal production plan. Additionally, we evaluated the influence of different ratios of orders with fixed deadlines on the performance of the algorithm. The used algorithm can successfully solve also these additional constraints.

1 INTRODUCTION

Complex manufacturing processes are needed to produce various types of products. Often, each type of similar product requires different steps, and different product parts for completion. The effective production plan has to allow on-time delivery, while minimizing production costs in terms of overall production time. Additionally, sometimes there are some specific constraints to be considered, i.e., orders with fixed deadlines. These orders have to be produced at the exact date, which causes some other orders to be produced either too early or with the delay. In any case, the main problem in production processes is the exchange delay, caused by adapting production lines to different types of products and supplying the appropriate parts, when switching from one type to another on the same production line.

Several useful and efficient scheduling methods are reported in the literature, including a genetic algorithm (GA) and its variants. (Senthilkumar and Shahabudeen, 2006) developed a heuristic for the open job shop scheduling problem using GA to minimize makespan, while (Chryssolouris and Subramaniam, 2001) developed a scheduling method based on GA and considering multiple criteria. Some other implementations of GA for job scheduling is reported by (Vazquez and Whitley, 2000). One of the advanced optimization techniques is a family of memetic algorithms (MAs) (Ong and Keane, 2004), which represent a synergy of evolutionary approach with separate individual learning or local improvement procedures for problem search. Various MAs were developed (Caumond et al., 2008; Hasan et al., 2009) to obtain even better results than GA. The results of MAs not only improved the quality of solutions but also reduced the overall computational time (Hasan et al., 2009). Furthermore, several researchers try to develop an algorithm that would be able to solve the problem without human intervention for setting the suitable control parameters (Brest et al., 2006; Harik and Lobo, 1999). In this paper we checked the influence of the orders with fixed deadlines on the performance of the advanced parameter-less implementation of GA. The problem and its solving with MA was initially introduced in (Korošec et al., 2010).

The rest of the paper is organized as follows. In Section 2 the problem is formally defined; in Section 3 the search approach with parameter-less algorithm is described; in Section 4 the experimental environment and results are presented; and in Section 5 some conclusions are listed.

2 PRODUCTION PLANNING

Our production planning problem can be represented as a job scheduling problem, which is NP-hard (Brucker, 1998). A schedule is an allocation of one or

Vukašinovič V., Korošec P. and Papa G..
HYBRID PARAMETER-LESS EVOLUTIONARY ALGORITHM IN PRODUCTION PLANNING.
DOI: 10.5220/0003085002310236
In *Proceedings of the International Conference on Evolutionary Computation* (ICEC-2010), pages 231-236
ISBN: 978-989-8425-31-7

more time intervals for each job to one or more machines. Let us assume we have a finite set of n jobs, where each job consists of a chain of operations. The jobs in our production planning problem correspond to orders (with list of required products) and different operations correspond to different products required by the order. Then, we have a finite set of m machines, where each machine can handle at most one operation at a time. Machines correspond to production lines. Each operation needs to be processed during an uninterrupted period of a given length on a given production line. The objective is to find a schedule satisfying certain restrictions, while minimizing the overall execution time, i.e., time for execution of all operations. In our planning problem each production line has its own time schedule. Further, each order has its own deadline, which should not be missed, but can be executed anytime before the deadline. Each product can be done only on some production lines and on each of them the execution time is different. Changing the manufacturing process from one product to another may cause an exchange delay which depends also on the used production line. There is also a stock. If an operation consists of product that is in the stock then the operation consists of using the stock and producing only missing quantity of the product. There are several orders with fixed deadlines, which need to be produced on the exact day, but not before.

2.1 Mathematical Formulation of the Problem

Our production planning problem is defined by a set of orders $J = \{j_1, j_2, \dots, j_n\}$ and a set of production lines $M = \{m_1, m_2, \dots, m_m\}$, where the orders have to be processed. Each order j_i consists of a set of q_i operations $O_i = \{(o_{i1}, \tau_{i1}(m)), (o_{i2}, \tau_{i2}(m)), \dots, (o_{iq_i}, \tau_{iq_i}(m))\}$, where o_{ik} is an operation and $\tau_{ik}(m)$ is its processing time depended on production line used for $k \in \{1, 2, \dots, q_i\}$. Each operation consists of only one type of product, therefore its processing time equals processing time of the product p_{ik} on the production line used times number of p_{ik} products. We denote a finishing time of order j_i by F_i and its deadline by D_i. Every operation o_{ik} of order j_i has deadline D_i and finishing time $F(o_{ik})$. Further, we denote $exd(l, o_{i_1k_1}, o_{i_2k_2})$ exchange delay between the products of operations $o_{i_1k_1}$ and $o_{i_2k_2}$ on production line m_l. Notice, that the order of $o_{i_1k_1}$ and $o_{i_2k_2}$ is important. By $S[p_{ik}]$ we denote the number of products p_{ik} available in the stock.

Let us assume that N is the number of operations:

$$N = \sum_{k=1}^{n} |O_k|.$$

A schedule is denoted by

$$C = g_{11}g_{12}g_{21}g_{22} \cdots g_{N1}g_{N2}, \tag{1}$$

where g_{k1} is an index on some operation and g_{k2} is the production line used to perform operation g_{k1}, for $k \in \{1, 2, \dots, N\}$. The task is to find the schedule, that minimizes the number of delayed orders, exchange delays and time to finish all the orders. The number of delayed orders n_{orders} is

$$n_{\text{orders}} = \sum_{i=1}^{n} \text{delay}_i,$$

where

$$\text{delay}_i = \begin{cases} 0 & D_i - F_i \geq 0 \\ 1 & D_i - F_i < 0. \end{cases}$$

The overall exchange delay t_{exd} is

$$t_{\text{exd}} = \sum_{l=1}^{m} \sum_{i=1}^{N} \sum_{j=i+1}^{N} \delta_{l,i}\delta_{l,j}\, exd(l, g_{i1}, g_{j1}),$$

where

$$\delta_{l,i} = \begin{cases} 1 & g_{i2} = l \pmod{m} \\ 0 & \text{otherwise.} \end{cases}$$

The time to finish all the orders t_{all} is

$$t_{\text{all}} = \max_{i=1}^{n} F_i.$$

The number of days of delayed orders n_{days} are

$$n_{\text{days}} = \sum_{i=1}^{n} \text{delay}_i \lceil (F_i - D_i)/(24 \cdot 60) \rceil.$$

The constraints which also have to be considered are the following. For the process of every o_{ik} only some production lines are appropriate. We denote $\delta_{ik} = (\delta_{m_1}, \delta_{m_2}, \dots, \delta_{m_m})$, where

$$\delta_{ik}[l] = \delta_{m_l} = \begin{cases} 1 & o_{ik} \text{ can be done on line } m_l \\ 0 & \text{otherwise.} \end{cases}$$

Some of the orders have to be finished on exact day:

$$F_{i_l} = D_{i_l},$$

for some $j_{i_1}, j_{i_2}, \dots, j_{i_k} \in J$ and $l \in \{1, 2, \dots, k\}$.

The aim is to achieve $\sum_{i=1}^{n} \text{delay}_i = 0$, but we still allow schedule with delayed orders. The aim is also to achieve production line equilibration, which means we would like to minimize the finishing time difference between lines in M:

$$t_{\text{eq}} = \min \sum_{l=1}^{m} (t_{\text{all}} - \text{FM}_l),$$

where $\text{FM}_l = \max_{i=1}^{N}(\delta_{l,i} F(g_{i1}))$ is maximal finishing time on line m_l.

3 HYBRID ALGORITHM

We hybridized the Parameter-Less Evolutionary Search (PLES) (Papa, 2008) with the local search. The PLES is based on basic GA (Bäck, 1996; Goldberg, 1989), except it does not need any control parameter, e.g., population size, number of generations, probabilities of crossover and mutation, to be set in advance. They are set virtually, according to complexity of the problem and according to statistical properties of the solutions found. In its search process PLES tries to efficiently explore the whole search space in order to find the optimal solution. The hybridized algorithm (PLES+LS) possesses the ability of PLES to find a near-optimal solution in a reasonable time, and the power of local search to move quickly towards the optimal one.

```
HybridizedAlgorithm {
  SetInitialPopulation(P)
  Evaluate(P)
  Statistics(P)
  while (not EndingCondition()){
    ForceBetterSolutions(P)
    MoveSolutions(P)
    LocalSearch()
    Evaluate(P)
    Statistics(P)
  }
}
```

3.1 Population Initialization and Termination Criterion

The production schedule was encoded into one chromosome with a tuples of values, where each tuple (gene) consisted of the index of the enumerated order and the production line. Based on the given list of all orders, which are sorted according to the deadlines, various orders of indexes that represent the given order are encoded in chromosome. A chromosome, which includes encoded production schedule of n orders, as presented in Equation 1.

The initial population P consists of *PopSize* chromosomes. In each chromosome the orders are randomly distributed, and also the assigned production line is chosen randomly among possible lines for each order. Since the numbers in the chromosome represent the indexes of orders their values can not be duplicated and no number can be missed; therefore both conditions must be considered during the initialization.

The initial population size (*PopSize*) is set according to the following equation

$$PopSize = n + 10log_{10}(\sum_{i=1}^{m} lines_i)$$

where $lines_i$ is the number of possible lines of the i-th order.

The EndingCondition() function checks if there was no improvement for several generations; then the system was assumed to be in a steady state, and the optimization ended. The number of generations depends on the convergence speed of the best solution found. Optimization is running while a better solution is found every few generations. But when there is no improvement of the best solution for a few generations (*Resting*), the optimization process stops. The *Limit* (i.e., number of generations since the last improvement) for stopping the optimization process is defined as follows

$$Limit = 10log_{10}(PopSize) + log_{10}(Resting + 1)$$

where *Resting* is the number of generations since the last improvement of the global best solution.

3.2 Variable Population Size

During the search process the population size is changed every few generations ($\frac{Limit}{5}$), based on the average change of the standard deviation of solutions over a last few generations. When the standard deviation increases than the population size is decreased, and vice-versa. When the population is shrunk the randomly chosen solutions are removed from the population, and when the population is inflated, some randomly chosen solutions are duplicated.

$$PopSize_i = \frac{PopSize_{i-1}}{Change},$$

where *Change* is calculated as

$$Change = \frac{StDev_i + StDev_{i-1}}{StDev_{i-1} + StDev_{i-2}}.$$

Moreover the *PopSize* change is limited to 25% per change and is further limited to $[\frac{PopSize}{5}, 1.1\,PopSize]$.

3.3 Force Better Solution

In every generation worse solutions are replaced with better solutions, and up to 25% of genes in the chromosomes are switched. This operator incorporates (1) the function of elitism, while forcing to replace worse solutions with better ones, and (2) the function of crossover, while taking the good solutions and slightly change them on some positions.

3.4 Solution Moving

Typically, every chromosome is the subject of mutation in the basic GA. In PLES, mutation is performed through the moving of some positions in the chromosome according to different statistical properties.

First, only the solutions that were not moved within the "Force better" operator are handled here. In other words, solutions of the previous generation that were better than the average are moved. The number of the positions in the chromosome (*Ratio*) to be moved is calculated on the basis of the standard deviation of the solutions in the previous generation and the maximal standard deviation as stated in the following equation.

$$Ratio_i = \tanh\left(1 - \frac{StDev_{i-1}}{StDev_{max}}\right) \times N$$

where $StDev_{i-1}$ and $StDev_{max}$ are the standard deviation of the solution fitness of the previous generation, and the maximal standard deviation of all generations, respectively. Here $Ratio \in \{0 \ldots N\}$, and the *Ratio* positions in the chromosome are selected to be moved. The moves are implemented by changes of production lines and/or by switching the positions of genes in the chromosome.

3.5 Solution Evaluation and Statistics

Each population is statistically evaluated. Here the best, the worst, and the average fitness value in the generation are found. Furthermore, the standard deviation of fitness values of all solutions in the generation, the maximal standard deviation of fitness value over all generations, and the average value of each parameter in the solution are calculated.

3.6 Local Search

Local search was implemented with four procedures, which run sequentially.

- **Stock Replacing.** For each order filled from the stock it is checked, if some other order of the same product is scheduled for the production. If the second one is delayed, than it is shifted in front of the order from the stock. In this case some orders of the same product are possibly moved out of the stock and placed into the production. If the number of the delayed orders increases, then the shifted order is returned to its previous position.
- **Deadline Sorting.** For each production line, all orders with delayed deadlines are checked if they can be moved before some other order. The delayed order is moved before each of the precedent order, so that it is not delayed anymore, while ensuring that the total number of delayed orders is not increased.
- **Production Line Changing.** For each production line, all orders are checked if they can be placed on any other feasible production line. If they can be placed on some other production line, then it is further checked if they can be merged with some other similar order on that new production line. The switch to another production line should not increase the exchange delay on the new production line.
- **Similar Product Merging.** For each production line, it is checked if several orders can be merged together. The merging is performed in four steps, according to different properties of the products. First the orders for the products with the same height are merged, then those with the same size are merged, after that the orders with the same connectors, and finally those with the same power characteristics. The idea of merging procedure is to decrease the production time on each line, as result of decreased exchange delay on the line.

3.7 Fitness Evaluation

Each new solution in the population was evaluated, according to the number of delayed orders (n_{orders}), exchange delay times in minutes (t_{exchange}), overall production time in minutes (t_{overall}), and the number of days of delayed orders (n_{days}). The cost function, which is calculated inside Evaluate(P), is as follows:

$$f(P) = 10^7 \cdot n_{\text{orders}} + 10^4 \cdot t_{\text{exchange}} + t_{\text{overall}} + n_{\text{days}}^2.$$

According to the cost function it is obvious that the most important item to minimize is n_{orders}, then t_{exchange} and lastly t_{overall} and n_{days}. The weights of these items make sure that the first two digits of evaluation function value represent number of delayed orders, next three digits represent exchange delay times in minutes and the last digits represent the influence of t_{overall} and n_{days}.

4 PERFORMANCE EVALUATION

4.1 The Experimental Environment

The computer platform used to perform the experiments was based on AMD Athlon II ™ 2.9-GHz processor, 4 GB of RAM, and the Microsoft® Windows® 7 operating system. The PLES+LS was implemented in Sun Java 1.6.

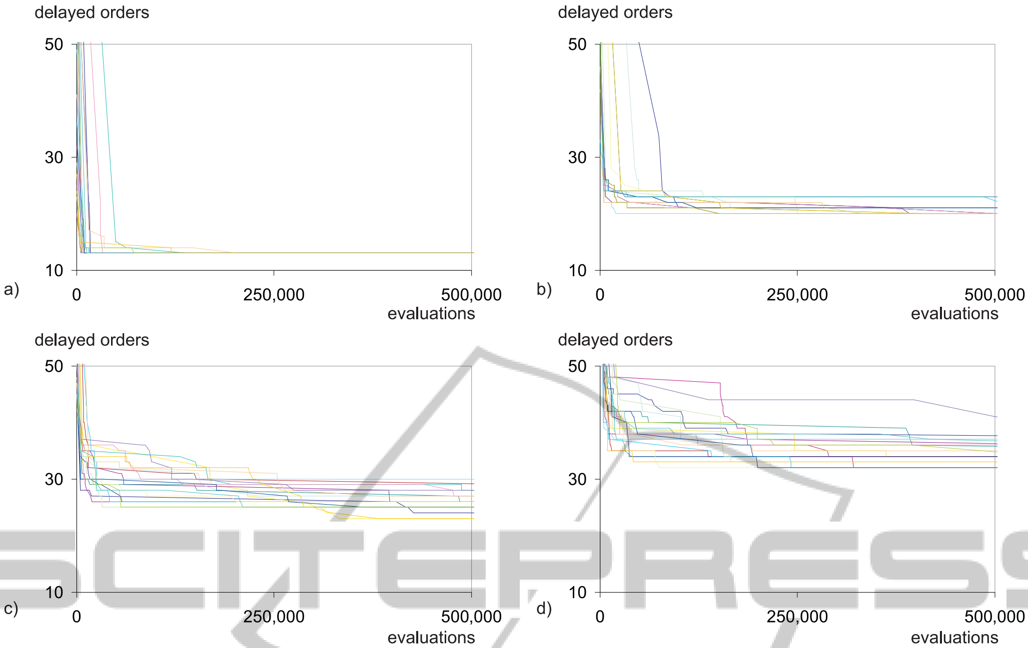

Figure 1: Influence of fixed orders on n_{orders} in Task 1: a) 0%, b) 5%, c) 10%, d) 25%.

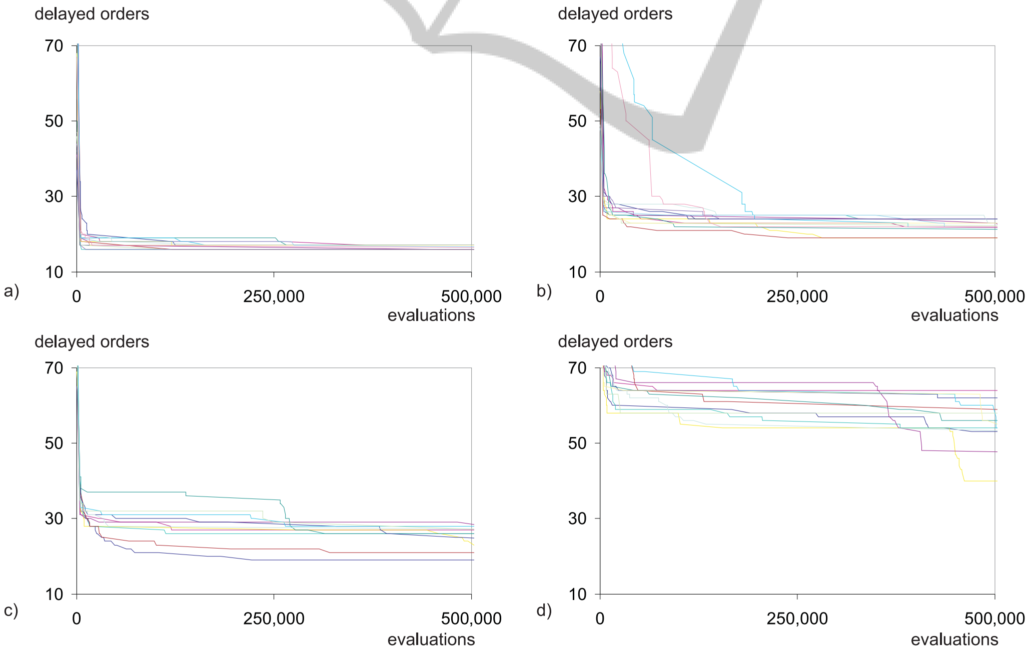

Figure 2: Influence of fixed orders on n_{orders} in Task 2: a) 0%, b) 5%, c) 10%, d) 25%.

4.2 The Test Cases

The PLES+LS algorithm was tested on two different real order lists from production company. The first task (Task 1) consists of $n = 711$ orders for 251 products, while the second task (Task 2) consists of $n = 737$ orders for 262 products. In both tasks the ratio of orders with fixed deadlines varied (0%, 5%, 10%, 25%). In both tasks $m = 5$ production lines are available. Each product can only be put on some production lines (depending on the product characteristics). We made 30 runs for each task, while the number of evaluations was limited to $1,000,000$.

4.3 Parameter Settings

As stated before, the control parameters are never set in advance and are not constant. They are determined each time on the basis of statistical properties of each population. Besides the population size changes through the search progress - to enable the search with different population sizes.

Table 1: Results of optimization for Task 1.

	0%	5%	10%	25%
Best	1.314×10^8	2.017×10^8	2.316×10^8	3.227×10^8
Mean	1.319×10^8	2.037×10^8	2.561×10^8	3.417×10^8
Worst	1.325×10^8	2.117×10^8	2.819×10^8	3.738×10^8
StD	3.495×10^5	4.039×10^6	1.531×10^7	1.821×10^7

Table 2: Results of optimization for Task 2.

	0%	5%	10%	25%
Best	1.616×10^8	1.934×10^8	1.837×10^8	4.054×10^8
Mean	1.663×10^8	2.149×10^8	2.381×10^8	5.051×10^8
Worst	1.723×10^8	2.421×10^8	2.833×10^8	6.249×10^8
StD	4.814×10^6	1.592×10^7	3.744×10^7	5.836×10^7

Table 3: Comparison of delayed orders.

	Task 1				Task 2			
	0%	5%	10%	25%	0%	5%	10%	25%
Best	13	20	23	32	16	19	18	40
Mean	13	20	25	34	16	21	23	50
Worst	13	21	28	37	17	24	28	62

4.4 Results

In Tables 1 and 2 best, mean, worst, and standard deviations of solutions are presented for each ratio of fixed deadlines.

To show how some of the components of the cost function behave during the search process, we present Figures 1 and 2. It can be visually seen, how the number of orders with fixed deadlines influences the performance of PLES+LS. Here, each line represent one run. The number of delayed orders is increasing with the growing ratio of fixed-deadline orders.

When comparing with the previous approach (Korošec et al., 2010) of production planning, the influence of fixed orders is seen on the number of delayed orders. The details are presented in Table 3.

5 CONCLUSIONS AND FUTURE WORK

In this paper, we have shown an application of specialized hybrid parameter-less evolutionary algorithm on a real-world production planning problem. The presence of orders with fixed deadlines influences the performance of the hybrid PLES+LS algorithm. However, even at 25% of orders with fixed deadlines, the results are still better than those provided by the expert (Korošec et al., 2010).

REFERENCES

Bäck, T. (1996). *Evolutionary Algorithms in Theory and Practice*. Oxford University Press.

Brest, J., Mernik, S. G. B. B. M., and Žumer, V. (2006). Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems. *IEEE Transactions on Evolutionary Computation*, 10(6):646–657.

Brucker, P. (1998). *Scheduling algorithms*. Springer, Heidelberg, 2nd edition.

Caumond, A., Lacomme, P., and Tchernev, N. (2008). A memetic algorithm for the job-shop with time-lags. *Comput. Oper. Res.*, 35(7):2331–2356.

Chryssolouris, G. and Subramaniam, V. (2001). Dynamic scheduling of manufacturing job shops using genetic algorithms. *Journal of Intelligent Manufacturing*, 12(3):281–293.

Goldberg, D. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley.

Harik, G. and Lobo, F. (1999). A parameter-less genetic algorithm. In *Proc. Genetic and Evolutionary Computation Conference (GECCO 1999)*, pages 258–265.

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

Korošec, P., Papa, G., and Vukašinović, V. (2010). Application of memetic algorithm in production planning. In *Bioinspired Optimization Methods and their Applications*, pages 163–175.

Ong, Y. and Keane, A. (2004). Meta-lamarckian learning in memetic algorithms. *IEEE Transactions on Evolutionary Computation*, 8(2):99–110.

Papa, G. (2008). Parameter-less evolutionary search. In *Proc. Genetic and Evolutionary Computation Conference (GECCO'08)*, pages 1133–1134.

Senthilkumar, P. and Shahabudeen, P. (2006). Ga based heuristic for the open job shop scheduling problem. *The International Journal of Advanced Manufacturing Technology*, 30(3-4):297–301.

Vazquez, M. and Whitley, L. D. (2000). A comparison of genetic algorithms for the static job shop scheduling problem. In *Parallel Problem Solving from Nature*, pages 303–312. Springer.