Simulation in Produktion
und Logistik 2017
Sigrid Wenzel & Tim Peter (Hrsg.)
kassel university press, Kassel 2017

A Two-phase Genetic Algorithm to Solve a Multi-objective Problem for Complex Assembly Lines

Ein zweiphasiger genetischer Algorithmus zur Lösung eines Problems mit mehreren Zielen für komplexe Montagelinien

Zhugen Zhou, Evangelos Angelidis, Daniel Bohn, Oliver Rose, Universität der Bundeswehr München, München (Germany), zhugen.zhou@unibw.de, evangelos.angelidis@unibw.de, daniel.bohn@unibw.de, oliver.rose@unibw.de

Abstract: Many real engineering problems involve multiple objectives, i.e., maximize factory performance and minimize production cost. In this paper, we focus on multi-objective optimization for complex assembly lines with workforce constraints. In these production systems, each order has its own production objectives, e.g., individual cycle time, tardiness and cost. Meanwhile, overall performance of the system, e.g., total cycle time, tardiness and cost, is the focus as well. A practical problem is to optimize individual objectives of different orders under conflicting situation, while keeping total objectives of the system in acceptable levels. To tackle this problem, we develop a multi-objective genetic algorithm combining with a two-phase altering objective functions procedure. Simulation experiments on a printing machine production model show its effectiveness of solving the problem in comparison with the peer algorithms.

1 Introduction

The research fields in which we operate are complex assembly lines for airplanes, printing machines, turbines and other heavy machines. In our past studies (Angelidis et al. 2013), a project network was used to represent this kind of production system. In the project network, each project that is composed of activities with precedence constraints denotes a production plan of product. This project network is highlighted by various modes of processing time and limited multi-skilled resources. This kind of production is defined as Multi-Mode Resource-Constrained Multi-Project Scheduling Problem (MMRCMPSP) that is an extension of conventional flow shop scheduling. Many optimization strategies in literature concentrate on single objective optimization for the MMRCMPSP (Gen and Cheng 1997). As a matter of fact, multi-objective must be taken into consideration in real engineering situation, in particular, in the complex assembly lines. One characteristic of this project network is each project representing an order or a product has its own performance measure named individual objective, e.g., cycle time, tardiness and cost, because of the product customization. Due to fierce

competition for the limited resources, optimizing a particular objective for a project can result in unacceptable results to other projects. Therefore, it is necessary to search the compromise solutions among the conflicting objectives. Meanwhile, overall performance of the project network called total objective, e.g., total cycle time, total tardiness and total cost, is of interest as well. A practical problem arising here is, on one hand, we intend to optimize the individual objectives of different projects that are under conflicting situation. On the other hand, we expect that the total objectives of the project network are controlled in desired levels. Assuming there are three orders, and the cycle time of each is the objective. In the first front, there are three non-dominated solutions *A* (90, 40, 50), *B* (50, 90, 100) and *C* (120, 80, 30). These three solutions are considered as good solutions as they are non-dominated with each other. However, when the total cycle time performance is expected to be lower than 200, solution *A* (180) is the only eligible one because *B* is 240 and *C* is 230. It is of importance that providing good and eligible solutions as many as possible in a short time, which provides different types of options for the decision makers to make trade-off or choose solutions they prefer. Although there has been an increase in the multi-objective optimization using genetic algorithms (MOGA) in the last decades, the majority of the literature, for instance, random weight MOGA (Murata et al. 1996), two hybrid GAs (Cavalieri and Gaiardelli 1998), cellular MOGA (Murata et al. 2001), non-dominated sorting generic algorithm II (NSGA-II) (Deb et al. 2002) and Lorenz NSGA (L-NSGA) (Dugardin et al. 2010), is to deal with the overall performance of the project network, and there is still a lack of satisfactory solution to the problem we concern.

In this paper, we develop a MOGA consisting of elitist method and diverse population as other MOGAs do to solve MMRCMPSP for multi-objective optimization. In particular, based on this MOGA, we develop a two-phase altering objective functions procedure (two-phase MOGA) to improve the situation that simultaneous optimization for individual objectives may bring conflict to the desired total performance. As we already indicated, the problem we are facing is a large real engineering problem. Taking computational feasibility into consideration, instead of exploring the entire Pareto optimal set, we concentrate in searching non-dominated solutions as many as possible in a short time.

2 Multi-objective Genetic Algorithm

The MOGA comprises two parts which are multi-objective (fitness assignment) and genetic algorithm (key components listed in Table 2).

2.1 Multi-objective and Fitness Assignment

In this research, we target at three objectives minimization for the order (project) level and the production system (project network) level. Cycle time (the difference between release time and finish time), tardiness (the difference between due date and finish time) and cost (the cost for the usage of resources to perform activities) are considered as performance measures. For the order level, as the simulation model contains three orders (Section 3), three cases for individual objectives optimization are designed and displayed in Table 1. For the production system level, total cycle time (the sum of cycle time of all orders), total tardiness (the sum of tardiness of all orders) and total cost (the sum of cost of all orders) are three considered objectives. To evaluate the

fitness of a solution, based on Pareto dominance concept (Goldberg 1989) we apply the fast non-dominated sorting algorithm (Deb et al. 2002) which assigns a rank to a solution, rather actual objective function value.

Table 1: *Three objectives optimization for order and production system*

		Objective 1	Objective 2	Objective 3
Order	Case 1	$\text{CycleTime}_{\text{order\_1}}$	$\text{CycleTime}_{\text{order\_2}}$	$\text{CycleTime}_{\text{order\_3}}$
	Case 2	$\text{Tardiness}_{\text{order\_1}}$	$\text{Tardiness}_{\text{order\_2}}$	$\text{Tardiness}_{\text{order\_3}}$
	Case 3	$\text{Cost}_{\text{order\_1}}$	$\text{Cost}_{\text{order\_2}}$	$\text{Cost}_{\text{order\_3}}$
Production system		$\text{CycleTime}_{\text{production\_system}}$	$\text{Tardiness}_{\text{production\_system}}$	$\text{Cost}_{\text{production\_system}}$

2.2 Genetic Algorithm

Table 2: *Key components of the MOGA*

	Method	Detail
Chromo-some	Each chromosome is made of n+1 genes.	n is the number of activities of the project network, and the last gene (n+1) denotes the dispatching rule that generates priorities of the activities.
Popula-tion for-mation	A population consists of three parts - 'X', 'Y' and 'Z' (Fig. 1) (Kühn et al. 2016): 'X': comes from the non-dominated solutions set that contains the solutions from the first front to the fifth front discovered by the GA so far (elitism method). 'Y': is generated from the previous population via selection, crossover and mutation. 'Z': is created randomly to maintain diversity of the population. It is described by one parameter 'random_create_rate'.	For 'X': a parameter 'copy_rate_from_pareto' is used to represent percentage of the population copied from the non-dominated solutions. The MOGA applies tournament selection with a size of 2 to 'copy' the solutions from the non-dominated solutions set. If two solutions have different Pareto ranks, the one with smaller rank will be selected. Otherwise, the one with larger crowding distance (described in 'Solve genetic drift problem') will be selected, as shown in Algorithm 1.
Selection	Rank-based roulette wheel selection (Jadaan et al. 2005) is utilized to select solutions as parents to create offspring. To obtain the sorted solutions, we use two sorting procedures that consider Pareto rank as main sorting	Firstly, the solutions are sorted in ascending order based on their Pareto ranks. Then the solutions that have the same Pareto rank are sorted in descending order according to their crowding distances. Then, the mapping function that is used to

	and crowding distance as secondary sorting, as shown in Algorithm 2.	calculate the scaled ranks of solutions is shown in Equation (1).
Crossover	Employ parameterized uniform crossover to create offspring (Spears and DeJong 1991). After two solutions are selected as parents, at each gene a biased coin is tossed to determine which parent will pass the gene to the offspring.	It assumes that there are two parents A and B, and A has better fitness than B. A toss of head will choose the gene from parent A, and a toss of tail will select the gene from parent B. A parameter 'tail_rate' is used to represent the probability to toss a tail.
Solve genetic drift problem	The population tends to form relatively few clusters that prevent diverse populations, which is called genetic drift. Thus, the solutions located in densely populated areas should be penalized, which means their selection probabilities as parents should be reduced.	The first step is to calculate the Euclidean distance for every solution pair X and Y (Fonseca and Fleming 1993) (see Eq. 2). Then we calculate the crowding distance of solution X to other solutions in the population (see Eq. 3).
Solve optimi-zation conflict between individual and total objectives	The idea is the evolving generations are divided into two parts. The objective functions are switched from total objectives to individual objectives as the GA evolves from the first part generations to the second part generations. This approach increases the probability of a solution, if whose total performance is better than the average value of total performance of all the first front solutions, to be selected as parents. In such a way, it is expected to create non-dominated solutions for the conflicting individual objectives. In the meantime, the total performance can be controlled in desired level.	In the first phase, we consider the total performance of the production system, i.e., total cycle time, total tardiness and total cost, as objectives for the first part generations. The main purpose is to set up a basic search direction in which the total performance can be controlled within the expected range in the second phase. As long as the GA evolves to the second part generations, the objectives are switched to individual performance of the orders. At first the non-dominated solutions generated from the first phase are re-ranked by the fast non-dominated sorting algorithm according to the considered individual objectives. Via Equation (4) and Algorithm 3, we assign total performance ranks to solutions and sort population.

Algorithm 1: *Copy solutions from non-dominated solutions set*

```
If solution(i)Pareto_rank is not equal to solution(j)Pareto_rank
     Select min (solution(i)Pareto_rank, solution(j)Pareto_rank)
Else select max (solution(i)crowding_distance, solution(j)crowding_distance)
```

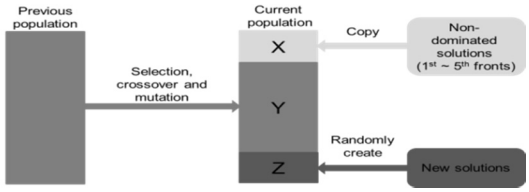

Figure 1: *Population formation*

Algorithm 2: *Sort population by Pareto rank and crowding distance*

```
Sort (sort (population) ascending by Pareto rank) descending
by crowding distance
```

$$\text{ScaledRank(pos)} = 2 - 2 * (\text{SP} - 1) * \frac{pos-1}{n-1} \quad (1)$$

Where *SP* is the selective pressure (1<SP≤2) that controls the bias between good and bad solutions, and it is considered as one parameter of the MOGA, *pos* is the position of the sorted solutions in the population, the fittest solution has *pos*=1 and the worst one has *pos*=*n*, *n* is the number of solutions in the population.

$$\text{EuclideanDistance(x, y)} = \sqrt{\sum_{k=1}^{K} \left(\frac{f_k(x) - f_k(y)}{f_k^{max} - f_k^{min}}\right)^2} \quad (2)$$

Where *k* is index of objective, f_k^{min} and f_k^{max} are the minimum and maximum value of the objective function $f(x)$ discovered so far, respectively.

$$\text{CrowdingDistance(x, S)} = \sum_{s=1}^{S} \text{EuclideanDistance(x, s)} \quad (3)$$

Where *s* is the index of solution in the population.

$$\text{Avg\_of\_first\_front} = \frac{\sum_{i=1}^{I} \sum_{j=1}^{J} \text{solution(i)}_{\text{individual\_objective(j)}}}{n} \quad (4)$$

Where *i* is the solution index in the first front, *j* is objective index, *n* is the number of solutions in the first front.

Algorithm 3: *Assign total performance rank to solution, and sort population by Pareto Rank, total performance rank and crowding distance*

```
If solution(i)total_performance is lower than the avg_of_first_front
      solution(i)total_performance_rank = 1
Else  solution(i)total_performance_rank = 2
Sort (sort (sort (population) ascending by Pareto rank)
ascending by total performance rank) descending by crowding
distance
```

2.3 Procedures of MOGA and MOGA Combining with a Two-phase Altering Objective Functions

The procedure of the MOGA is described as follows:

- Step 1: Generate a random population.
- Step 2: Assign a Pareto rank to each solution according to the fast non-dominated sorting algorithm. Then update the non-dominated solutions set.
- Step 3: Using selection, crossover and mutation to create new solutions as follows:
  - Step 3.1: Calculate crowding distances of solutions by Equation (2) and (3).
  - Step 3.2: Sort the population according to Algorithm 2.
  - Step 3.3: Calculate the scaled ranks of solutions according to Equation (1).
  - Step 3.4: Apply rank-based roulette wheel selection method to select solutions as parents.
  - Step 3.5: Apply uniform crossover method to create new solutions.
  - Step 3.6: Apply a mutation operator (probability of 0.05) to the new generated solutions.
- Step 4: Based on the new generated solutions, form a new population via copying solutions from the non-dominated solutions set by Algorithm 1 and randomly creating new solutions.
- Step 5: If stopping criteria is met, terminate the search, otherwise go to Step 2.

The MOGA combing with a two-phase altering objective functions procedure (two-phase MOGA) is given as follows:

- Phase 1:
  - Phase 1.1: Set total objectives to the production system, and number of generations for the Phase 1 optimization. After that, generate a random population.
  - Phase 1.2: Run the MOGA mentioned above from Step 2 to Step 4.
  - Phase 1.3: If the specified generations are reached, go to Phase 2, otherwise go to Phase 1.2.
- Phase 2:
  - Phase 2.1: Set individual objectives to the orders, and the number of generations for the Phase 2 optimization. Then, resort the non-dominated solutions set according to the individual objectives by the fast non-dominated sorting algorithm.
  - Phase 2.2: Run the MOGA from Step 2 to Step 4 with a replacement of Step 3.2. New step 3.2: Calculate the 'avg_of_first_front' from the non-dominated solutions set by Equation (4); then assign the overall performance ranks to the solutions and sort the population according to Algorithm 3.
  - Phase 2.3: If the specified generations are reached, terminate the search, otherwise go to Phase 2.2.

3 Simulation Experiment and Results Analysis

The case study is based on a complex assembly line for the production of printing machines. The model structure is from our industrial partner, and the data is artificial for this test case. In this production system, there are 3 orders (3 project networks) with a total of 12 products. The processing times of activities vary between 30 to 480

minutes. Each activity has different execution modes that rely on the amount and the required skills of resources. Each order has its own release date and due date. Table 3 lists the basic information of the orders. There are 18 different types of resources divided into 16 types of workers, e.g., electrician and fitter, and 2 types of processing facilities, e.g., crane. The workers are classified into 8 types of internal workers and 8 types of sub-contracted workers. The number of each resource is 3. Each resource is associated with a cost unit that is used to calculate the production cost of usage for the resource on activities. The interested readers can find more details about modeling for this kind of complex assembly in (Angelidis et al. 2013).

In this study, we apply simulation based optimization concept which is wildly used to obtain optimal solutions in large search space problem as the complex assembly line we focus on. A simulation based optimization platform (SBOP) which specializes in modeling, simulation and optimization is created (Angelidis et al. 2013). The optimization cycle of SBOP is shown in Figure 2. The first step is the transformation of the real system to the manufacturing model. The model describes all necessary production information as previously described. The optimizer in which our two-phase MOGA is implemented sets parameters and creates different simulation scenarios. After that, the simulation scenarios are simulated via the internal simulator and the simulation results are evaluated. This cycle is terminated when the target or the abort criteria is reached.

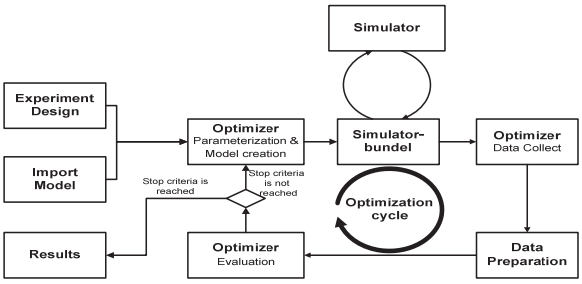

Figure 2: *Simulation based optimization platform (SBOP)*

As demonstrated in Table 2, 'copy_rate_from_pareto', 'random_create_rate', 'selective_pressure' and 'tail_rate' are considered as four parameters that potentially influence the search ability of MOGA. Before examining the performance of the two phase MOGA, we need to figure out the significance of these four parameters. Therefore, we carry out a design of experiment in which 3 levels of each parameter are defined and in total there are 81 combinations, as shown in Table 4. In this experiment, total cycle time, total tardiness and total cost are considered as three objectives. The population size is 50 and the evolution generations are 50, and the dispatching rule is FIFO (first in first out). Due to large amounts of data, the detailed results are not presented. Among this 82 parameter combinations, the one with 'copy_rate_from_pareto' (0.5), 'random_create_rate' (0.1), 'selective_pressure' (1.9), 'tail_rate' (0.5) is considered the best combination to produce the best result. This parameter combination is used for the following experiment.

Table 3: *Basic information of orders*

	Order 1			Order 2				Order 3				
Products	1	2	3	4	5	6	7	8	9	10	11	12
Activities	31	28	129	91	29	105	25	165	128	62	73	70

Table 4: *Design of experiment of the MOGA*

	Level 1 (low)	Level 2 (middle)	Level 3 (high)
copy_rate_from_pareto	0.1	0.3	0.5
random_create_rate	0.1	0.2	0.3
selective_pressure	1.1	1.5	1.9
tail_rate	0.3	0.5	0.7

We continue the experiment targeting at individual cycle time optimization which is the case 1 in Table 1 (as a preliminary study, the individual tardiness and cost optimizations are not included in this paper, only the results of individual cycle time optimization will be presented). As the orders are the focus, we expect to explore diverse solutions regarding different behaviours based on different dispatching rules. Therefore, FIFO, earliest due date (EDD, due date oriented) and shortest processing time (SPT, processing time oriented) are considered as three dispatching rules that are encoded as a gene in the chromosome. The population size is 50 and the evolution generations are 50. In the first phase, we employ total objectives for the first 25 generations and individual objectives for the rest 25 generations for the second phase. The random weight MOGA and NSGA-II are the benchmark algorithms. Table 5 lists three performance measures. From previous four parameters configuration experiment, we obtain the minimum total cycle time – 5,116 hours when the total performance is considered as objective. If we determine a good solution whose total cycle time is less than 5,116 hours, for random weight MOGA and NSGA-II, 6 out of 34 (18 %) and 11 out of 29 (38 %) solutions fulfil this requirement, respectively. However, our two-phase MOGA achieves 21 out of 38 (55 %) solutions. Comparing to these two peer algorithms, our two-phase MOGA is able to obtain more solutions whose total performance is controlled to the targeted level.

Figure 3 demonstrates the ratio of non-dominated individuals (RNI) (Tan et al. 2002), which proves our two-phase MOGA is able to produce more non-dominated solutions than random weight MOGA and NSGA-II. RNI is defined as: *RNI (X) = nondom_indiv / P*, where nondom_indiv is the number of non-dominated individuals (solutions) in population *X* and *P* is the size of population *X*. For instance, two sets of first front solutions from two-phase MOGA and NSGA-II are mixed. Then, the solutions which are non-dominated are selected to form a new population *X*. Finally, the RNI of two-phase MOGA is determined as the ratio of the number of non-dominated solutions (from two-phase MOGA) and the size of population *X*.

Table 5: *Three performance measures comparison, individual cycle time*

	Number of the first front solutions	Average value of total cycle time for all solutions in the first front (hours)	Number of the first front solutions whose total cycle time is less than 5116 hours
random weight MOGA	34	5542	6
NSGA-II	29	5386	11
Our two-phase MOGA	38	5105	21

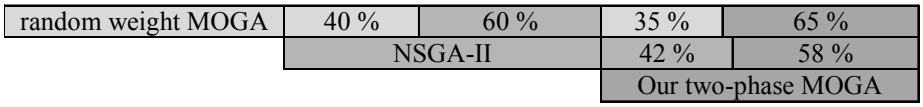

Figure 3: *RNI comparison among three algorithms, individual cycle time*

4 Conclusion

In this paper, we developed a MOGA to solve multi-objective optimization for complex assembly lines. Two kinds of performance measures, which are individual objectives for orders and total objectives for production system, were of interest. A printing machine production model was utilized to test the performance of our MOGA. Due to confliction, optimization for individual objectives may bring unacceptable levels to total objectives. Thus, we developed a two-phase procedure to alter objective functions for our MOGA to solve this problem. The simulation results proved that compared to the counterparts, the two-phase MOGA was able to produce more solutions whose total performance is superior to the targeted level.

Indeed, one feature of our MOGA is we can investigate its search ability by parameters configuration. We believe there will be room for improvement if we carry out an in-depth research for significance of parameters, which is considered as a possibility of future research. Moreover, the two-phase altering objective functions procedure can be adapted and applied to other problem domains, as long as both individual performance and total performance are the focus. Also, we are interested in testing different sizes of problems considering other performance as objective like resource utilization, and applying more dispatching rules.

Acknowledgement

The presented work is a result of the research project "Simulationsbasierte dynamische Heuristik zur verteilten Optimierung komplexer Mehrziel-Multiprojekt-Multiressourcen-Produktionsprozesse" (Funded by the Deutsche Forschungsgesellschaft (DFG), Duration 04/2013-03/2017).

References

Angelidis, E.; Bohn, D.; Rose, O.: A simulation tool for complex assembly lines with multi-skilled resources. In: Pasupathy, R.; Kim, S.-H.; Tolk, A.; Hill, R.; Kuhl, M. E. (Eds.): Proceedings of the 2013 Winter Simulation Conference (WSC), Washington, D.C. (USA), 2013, S. 2577-2586.

Cavalieri, S.; Gaiardelli, P.: Hybrid genetic algorithms for a multiple-objective scheduling problem. Journal of Intelligent Manufacturing 9 (1998), pp. 361-367.

Deb, K.; Pratap, A.; Agarwal, S.; Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGAII. IEEE Transactions on Evolutionary Computation 6 (2002), pp. 182-197.

Dugardin, F.; Yalaoui, F.; Amodeo, L.: New Multi-objective method to solve reentrant hybrid flow shop scheduling problem. European Journal of Operational Research 203 (2010), pp. 22-31.

Fonseca, C.M.; Fleming, P.J.: Multiobjective genetic algorithm. In: IEE Colloquium on Genetic Algorithms for Control Systems Engineering, London (UK), 1993, pp. 1-6.

Gen, M.; Cheng, R.: Genetic algorithm and engineering design. New York: John Wiley and Sons 1997.

Jadaan, O.A.; Rajamani, L.; Rao, C.R.: Improved selection operator for GA. Journal of Theoretical and Applied Information Technology 4 (2008), pp. 269-277.

Kühn, M; Zahid, T.; Völker, M.; Zhou, Z. Rose, O.: Investigation of genetic operators and priority heuristics for simulation based optimization of multi-mode resource constrained multi-project scheduling problems (MMRCMPSP). In: Claus, T.; Herrmann, F.; Manitz, M.; Rose, O. (Eds): Proceedings of European Council for Modelling and Simulation, Regensburg (Germany), 2016.

Murata, T.; Ishibuchi, H.; Tanaka, H.: Multi-objective genetic algorithm and its applications to flowshop scheduling. Computers & Industrial Engineering 4 (1996), pp. 957-968.

Murata, T.; Ishibuchi, H.; Gen, M.: Specification of genetic search directions in cellular multiobjective genetic algorithms. In: Zitzler, E.; Deb, K.; Thiele, L.; Coello Coello, C.A.; Corne, D. (Eds.) First International Conference on Evolutionary Multi-Criterion Optimization. Springer, Lecture Notes in Computer Science Zurich (Switzerland), 2001, pp. 82-95.

Spears, W.M.; Dejong, K.A.: On the virtues of parameterized uniform crossover. In: Belew, R.K.; Booker, L.B. (Eds.): Proceedings of the Fourth International Conference on Genetic Algorithms, San Diego (USA), 1991, pp. 230-236.

Tan, K.C.; Lee, T.H.; Khor, E.F.: Evolutionary algorithms for multi-objective optimization: performance assessments and comparisons. Artificial Intelligence Review 17 (2002), pp. 253-290.