Comparison of Multi-objective Approaches to the Real-World Production Scheduling

Gregor Papa and Peter Korošec

Abstract 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. In the paper, memetic versions of three multi-objective algorithms with different approaches to problem solving are implemented. The customized reproduction operators and local search procedures are also used. These memetic algorithms are applied to real order-lists from a production company. It is shown that the multi-objective approaches are able to find high-quality solutions, also when quick respond is required to adapt to dynamic business conditions. According to the results it is concluded that for the two tested real-world problems the IBEA confirmed its superiority over the NSGA-II and SPEA2.

1 Introduction

In the past we have already successfully approached a production-scheduling problem with a single-objective optimization [13]. 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

G. Papa (✉) · P. Korošec
Computer Systems Department, Jožef Stefan Institute, Ljubljana, Slovenia
e-mail: gregor.papa@ijs.si

P. Korošec
e-mail: peter.korosec@ijs.si

G. Papa
Jožef Stefan International Postgraduate School, Ljubljana, Slovenia

P. Korošec
Faculty of Mathematics, Natural Sciences and Information Technologies,
University of Primorska, Koper, Slovenia

E. Minisci et al. (eds.), *Advances in Evolutionary and Deterministic Methods for Design, Optimization and Control in Engineering and Sciences*, Computational Methods in Applied Sciences 48, https://doi.org/10.1007/978-3-319-89988-6_27

had to consider a multi-objective approach [9].

There was some initial investigation performed on the usage of multi-objective approaches. In the previous work [9] we used the Indicator-Based Evolutionary Algorithm (IBEA) [16], since it nicely upgrades on our initial work when solving the single-objective scheduling problem [13]. In current work we further improve the findings presented previously [9] with the comparison of the Non-dominated Sorting Genetic Algorithm-II (NSGA-II) [5], Strength Pareto Evolutionary Algorithm 2 (SPEA2) [17], and IBEA. Following the IBEA's proven performance for more than three objectives [14], 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

The growing complexity of the real-world scheduling problems forced significant work to be devoted to the automation of scheduling and planning processes. Here, we often have to deal with very large search spaces, real-time performance demands, and dynamic environments [11]. Effective production scheduling solutions can result in reduction of personnel and production costs by minimizing machine idle time and increasing the number of on-time job deliveries [2].

The multi-objective optimization [4] is very common within the world of engineering problems. As this approach deals with multiple objectives it is also recognized in solving of planning and scheduling problems.

The 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. This hybrid approach not only improves the quality of the solutions, but it also reduces the overall computational time [8].

In our initial work [13] 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) we developed and used specialized local searches. They were guided with the genetic algorithm, parameter-less evolutionary search [12], 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 [9] 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. While in [13] we proved the suitability of the evolutionary approach to finding an optimum solution within a broad range of possible solutions, in [9] we presented some additional local search procedures, as well as we introduced the multi-objective approach, where the IBEA algorithm was used.

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: the 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 [5] is the second version of the 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 enhances 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 to the rank of the front and distance within the front.

3.2 Strength Pareto Evolutionary Algorithm 2

The SPEA2 [17] is one of the 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 calculated.

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. The density of an area of the Pareto front is estimated upon the Euclidean distance of the objective values between a given solution and the nearest neighbors of the solution. It iteratively fills the archive population with the candidate solutions in order of their fitness. The most similar solutions are truncated from the archive population. For selection of parents some classical GA selection method, such as binary tournament selection or random selection, is used. The crossover and mutation are performed as classical operators of the GA.

3.3 Indicator-Based Evolutionary Algorithm

The IBEA [16] is a multi-objective version of the 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, and the optimization goal is the identification of a pareto-set that minimizes an indicator function. Using the indicator concept no additional diversity-preservation mechanisms are required. It was demonstrated [16] that an indicator-based search can yield results that are superior to some other widely-used algorithms such as the improved SPEA2 and NSGA-II.

In a basic version of the IBEA, binary tournaments are used 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 production scheduling problem was introduced in the company Eta Cerkno d.o.o., which produces components for domestic appliances [9, 13]. The most demanding production stage is the production of cooking hot plates. The fabrication process for various components used in different types of plates is similar, however due to clients' different demands the models differ in size (i.e., height, diameter), connector type, and power characteristics (i.e., wattage). For their logistic reasons the clients group different models of plates within the same order, implying the same due-dates for different products. Therefore, the production of these order groups must be scheduled very carefully to fulfil all the demands (i.e., 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 [15]. 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 [13].

4.1 Production Schedule Encoding

The production schedule is encoded into a chromosome with tuples of values. Each tuple (gene) consists of the index of the enumerated order and the assigned production line. A chromosome with production schedule of n orders, is presented in Eq. 1.

$$C = g_{1o}g_{1l} \quad g_{2o}g_{2l} \quad \cdots g_{ko}g_{kl} \quad \cdots \quad g_{no}g_{nl}, \tag{1}$$

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

4.2 Population Initialization

All input orders that have to be processed are firstly sorted within the initial order list, according to their due-dates. Next, different chromosomes are constructed as variations of the initial list, where each variation of the indexes of orders is encoded as a separate chromosome. In each chromosome the orders are randomly distributed, and the assigned production line is chosen randomly from among the possible lines for each order. The created initial population P consists of N chromosomes.

As the numbers that are encoded in the chromosome represent the indexes of orders, their values cannot be duplicated and also all indexes must be included. Also the assigned values for the production line depend on the possible production lines for particular order. These conditions have to be considered during the initialization as well as during all the subsequent phases.

4.3 Reproduction Operators

An order-based crossover operator interchanges positions that store the ordered numbers within some range. It takes the random part of two parents, and with a probability

p_c 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. In our implementation four types of order-based crossover operators are used: order (OX) [10], cycle (CX) [10], partially-mapped (PMX) [10] and PTL [3] crossover. They are switched every 10 generations of the optimization process.

During the mutation process each value of the chromosome mutates with a mutation probability p_m. Five different types of mutation, which are described with more details in [9], are applied: Changing of the production line; Switching of two genes in the chromosome; Shifting of a gene into some new position; Replacing similar products; Merging of similar products. The first mutation type influences the second part of the gene (i.e., g_{kl}); the second mutation type influences the whole gene ($g_{ko}g_{kl}$); the remaining three mutation types influence only the first part of the gene (i.e., g_{ko}).

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 probabilities are decreased during the algorithm execution.

4.4 Fitness Evaluation

The solutions $p \in P$ of each generation are evaluated after the reproduction operators and local search procedures modify them. Each solution p defines its set of objective values n_{obj}, where objective values are defined as: 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, $k \in \{1, ..., n_{obj}\}$.

4.5 Ending Condition

In general the algorithm is run until the user stops the optimization process. To mimic overnight running, as it is used in real setting to form new production schedules, we decided to limit the number of evaluations to 300 million.

5 Memetic Algorithms

There are several approaches for implementing local search procedures. In our case we merged the presented NSGA-II, SPEA2 and IBEA algorithms to guide the local search procedures. The basic algorithms are implemented with the use of appropriate Java classes of the jMetal framework [6]. 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.

Algorithm 1 Generic multi-objective memetic algorithm

```
1: SetInitialPopulation(P)
2: Evaluate(P)
3: while not EndingCondition() do
4:    P' = Selection(P)
5:    Crossover(P', p_c)
6:    Mutation(P', p_m)
7:    Evaluate(P')
8:    LocalSearch(P')
9:    P = PopulationManagement(P ∪ P')
10: end while
```

As presented in Algorithm 1 the pseudocode of a generic multi-objective memetic algorithm with different base algorithms is very similar. The main difference is in the Evaluate() function, which implements various fitness calculations (like raw fitness, density information, quality indicator...), and in PopulationManagement() function, which implements various algorithm-specific procedures (like sorting, crowding distance, and truncation procedure).

6 Performance Evaluation

6.1 Experimental Environment

The experiments were performed on the computer platform that is based on an AMD Opteron ™ 2.2-GHz processor, with 16 GB of RAM, and the Microsoft® Windows® 8.1 operating system. The algorithms are implemented in Sun Java 1.7.

6.2 Test Cases

For the fair comparison with the results from [9] the algorithms were tested on the same two real order lists from the production company. Task 1 consisted of $n = 470$ orders for 189 different products and Task 2 consisted of $n = 393$ orders for 175 different products. The number of orders n represents the problem dimension. The number of available production lines is $m = 5$.

As a comparison also a single objective result is presented. It was obtained as described in [9], where we used a lexicographic evaluation—the number of delayed orders (n_{orders}) was set as the most important objective, followed by the required

number of workers (n_{workers}), the sum of delayed days for all the delayed orders (n_{days}), and the sum of the change-over downtime in minutes (t_{change}).

6.3 Control Parameter Settings

The control parameters were based on the previous setting as used in [9], to achieve as equal as possible conditions for all compared algorithms:

- the population size $N = 500$;
- the crossover probability $p_{\text{c}} = 0.5$;
- the mutation probabilities $p_{\text{m}_{\text{change}}} = 0.01$, $p_{\text{m}_{\text{switch}}} = 0.01$, $p_{\text{m}_{\text{shift}}} = 0.01$, $p_{\text{m}_{\text{randomize}}} = 0.05$ and $p_{\text{m}_{\text{merge}}} = 0.5$;
- the number of evaluations was 300 million.

The implementation of the crossover and mutation was the same for all algorithms. The algorithms differ in their specific implementations of fitness calculation, selection procedures and in progress of the solutions into the next generations (i.e., how the offspring/archive population is managed).

6.4 Results

Results of comparison of all three algorithms for both tasks are presented in Figs. 1, 2, 3 and 4 as well as in Tables 1 and 2. In all the figures the X axis represents the n_{orders} objective, the Y axis represents the n_{workers} objective, the Z axis represents the t_{change} objective and the color scheme represents the n_{days} objective.

Similarly, as presented in [9], Figs. 1 and 2 show the pareto front for Task 1 in 4D space from different perspectives for all three compared algorithms.

In Fig. 1 we can see that for the Task 1 the "nicest" pareto front in regard to the XY plane is returned by the IBEA algorithm, while both the SPEA2 and NSGA-II produce more wide spreaded pareto front. Also the minimum acquired values are much lower at the IBEA algorithm, followed by the NSGA-II and lastly the SPEA2 algorithm.

In Fig. 2 we can see that for the Task 1 the pareto front in regard to the XZ an YZ planes is pronouncedly divided into two parts by the IBEA algorithm, while both the SPEA2 and NSGA-II produce more even pareto front with some dislocated solutions. The main reason for this is that the IBEA was able to generate "nicer" pareto front on XY plane, with solution with lower Y values required much higher X, Z, and colored values. This indicates that the n_{workers} objective invertly influences other objectives. For X values we see that IBEA was able to find lower solutions, while for all other objectives the quality of solution is much closer.

In Table 1 we can see the performance of all three multi-objective algorithms for Task 1, and also the solution obtained by the single objective approach is presented

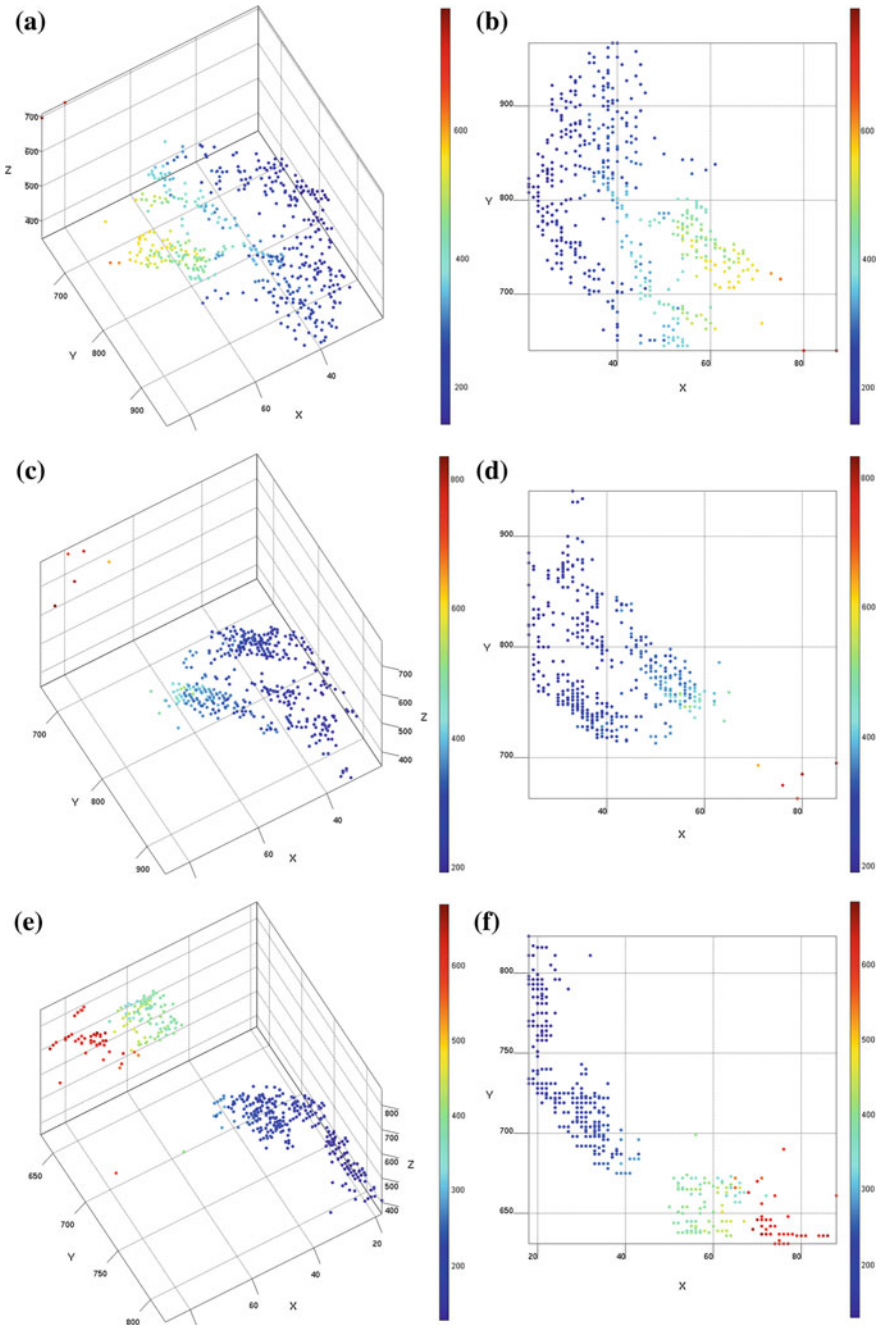

Fig. 1 Pareto front for Task 1 in the 4D space and XY plane: **a** NSGA-II 4D, **b** NSGA-II XY, **c** SPEA2 4D, **d** SPEA2 XY, **e** IBEA 4D, and **f** IBEA XY

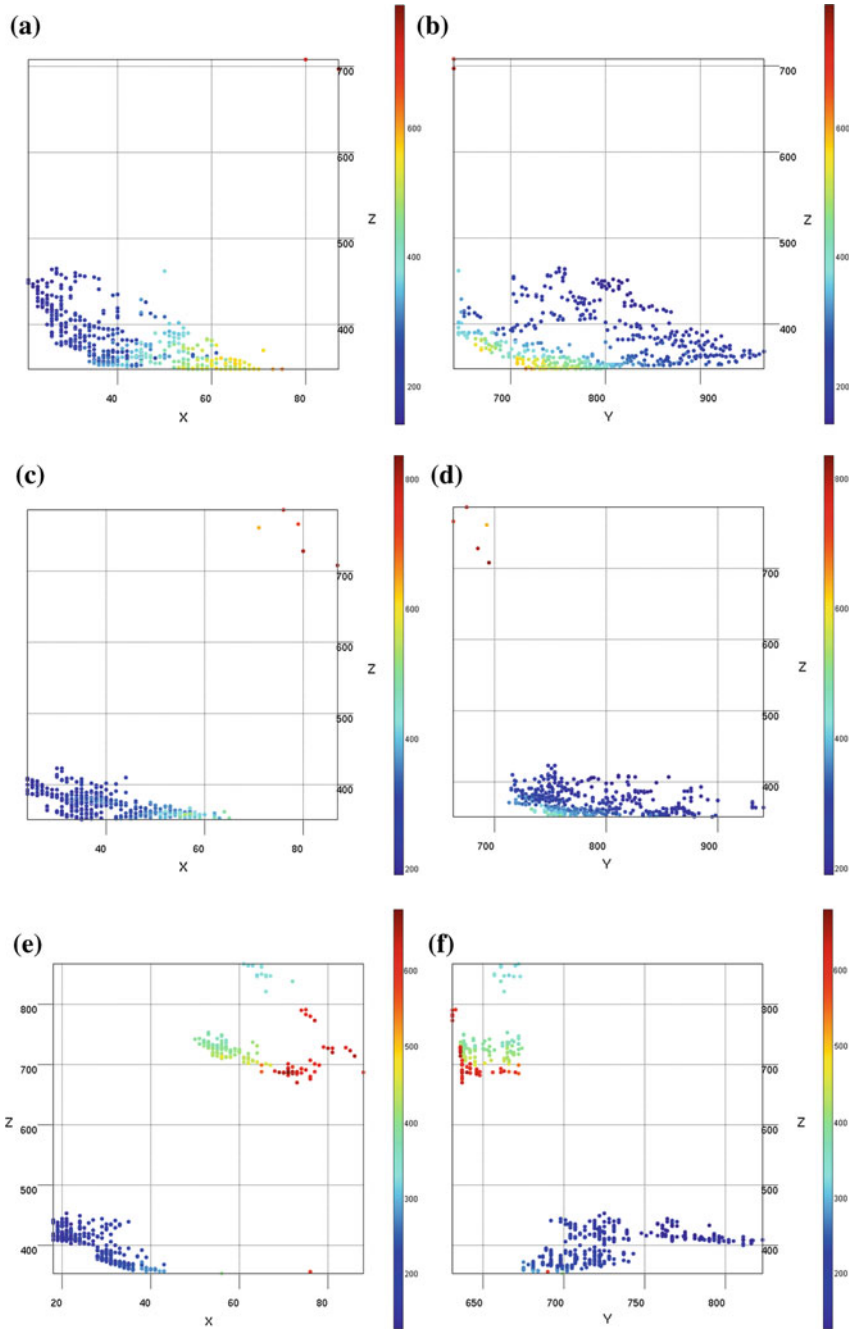

Fig. 2 Pareto front for Task 1 in the XZ and YZ plane: **a** NSGA-II XZ, **b** NSGA-II YZ, **c** SPEA2 XZ, **d** SPEA2 YZ, **e** IBEA XZ, and **f** IBEA YZ

Table 1 Results of optimization for task 1

Algorithm	Statistics	n_{orders}	n_{workers}	t_{change}	n_{days}
NSGA-II	pareto min	21	640	348	141
	Pareto max	87	967	708	790
	Pareto median	40	778	370	281
SPEA2	Pareto min	24	663	351	191
	Pareto max	87	941	786	836
	Pareto median	38	769	370	279
IBEA	Pareto min	18	631	353	127
	Pareto max	88	823	867	681
	Pareto median	32	704	414	206
Single-objective		18	767	714	156

Table 2 Results of optimization for Task 2

Algorithm	Statistics	n_{orders}	n_{workers}	t_{change}	n_{days}
NSGA-II	Pareto min	17	567	336	63
	Pareto max	56	902	655	621
	Pareto median	27	681	369	186
SPEA2	Pareto min	31	695	368	178
	Pareto max	82	999	451	1095
	Pareto median	39	779	385	284
IBEA	pareto min	16	538	355	59
	Pareto max	50	778	433	330
	Pareto median	26	601	371	101
Single-objective		15	702	443	155

as a comparison. The Table presents algorithms' pareto min, max, and median values for all objectives. The median value shows where the focus of search is. Since we are dealing with minimisation problem on all objectives, these are the values most interesting and indicating for us. When comparing the objective n_{orders} we can see that the range of values is more or less the same for all three algorithms, while the median value is the lowest at the IBEA. For the objective n_{workers} the IBEA has the smallest range and the lowest median value. For the objective t_{change} the IBEA has the largest range and a little bit higher median value than the other two algorithms. For the objective n_{days} the range of values is a little larger for the NSGA-II and SPEA2, while the median value of the IBEA is again the lowest one.

In Fig. 3 we can see that for the Task 2 the pareto fronts in regard to the XY plane are quite similar for the IBEA and NSGA-II , while SPEA2 produced much "weaker" pareto front (containing less solutions). Similarly to Task 1 the minimum acquired

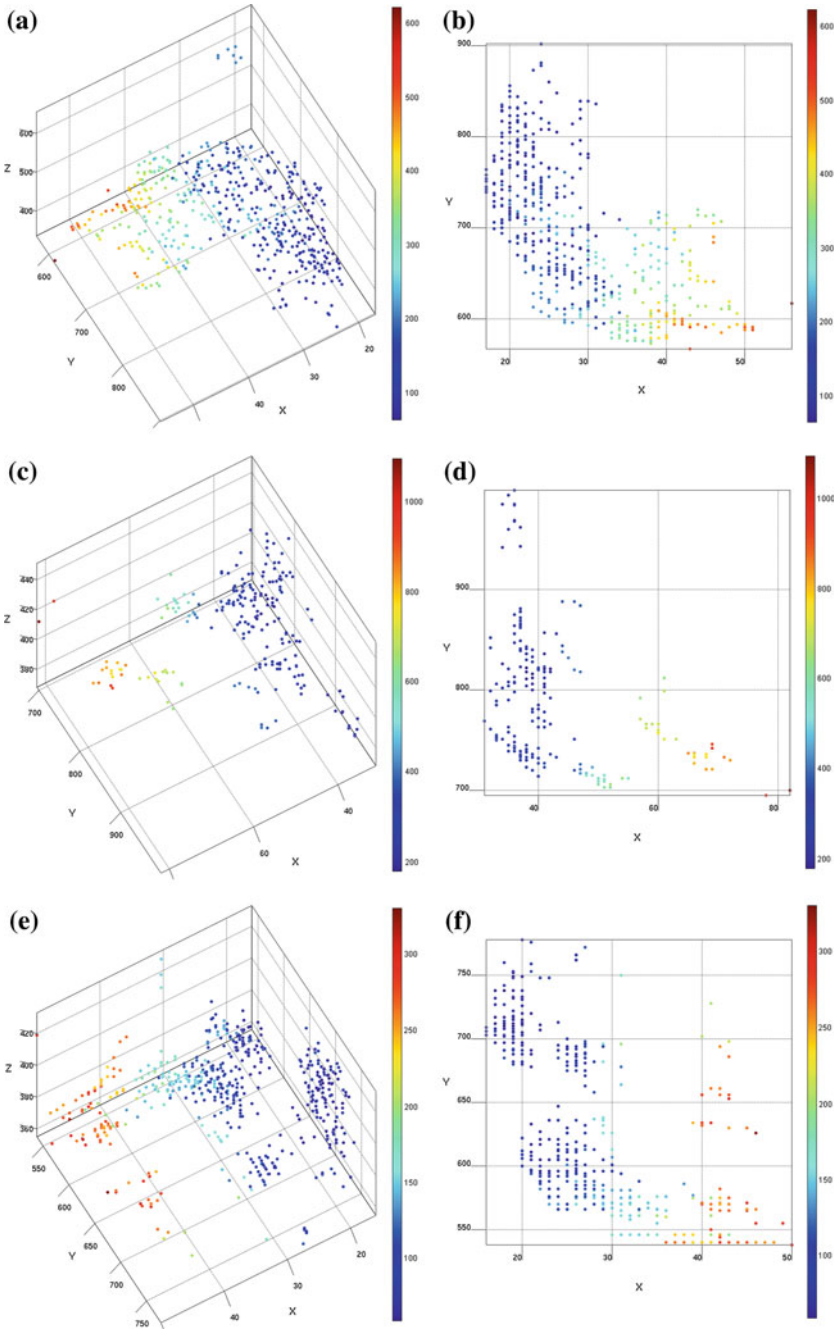

Fig. 3 Pareto front for Task 2 in the 4D space and XY plane: **a** NSGA-II 4D, **b** NSGA-II XY, **c** SPEA2 4D, **d** SPEA2 XY, **e** IBEA 4D, and **f** IBEA XY

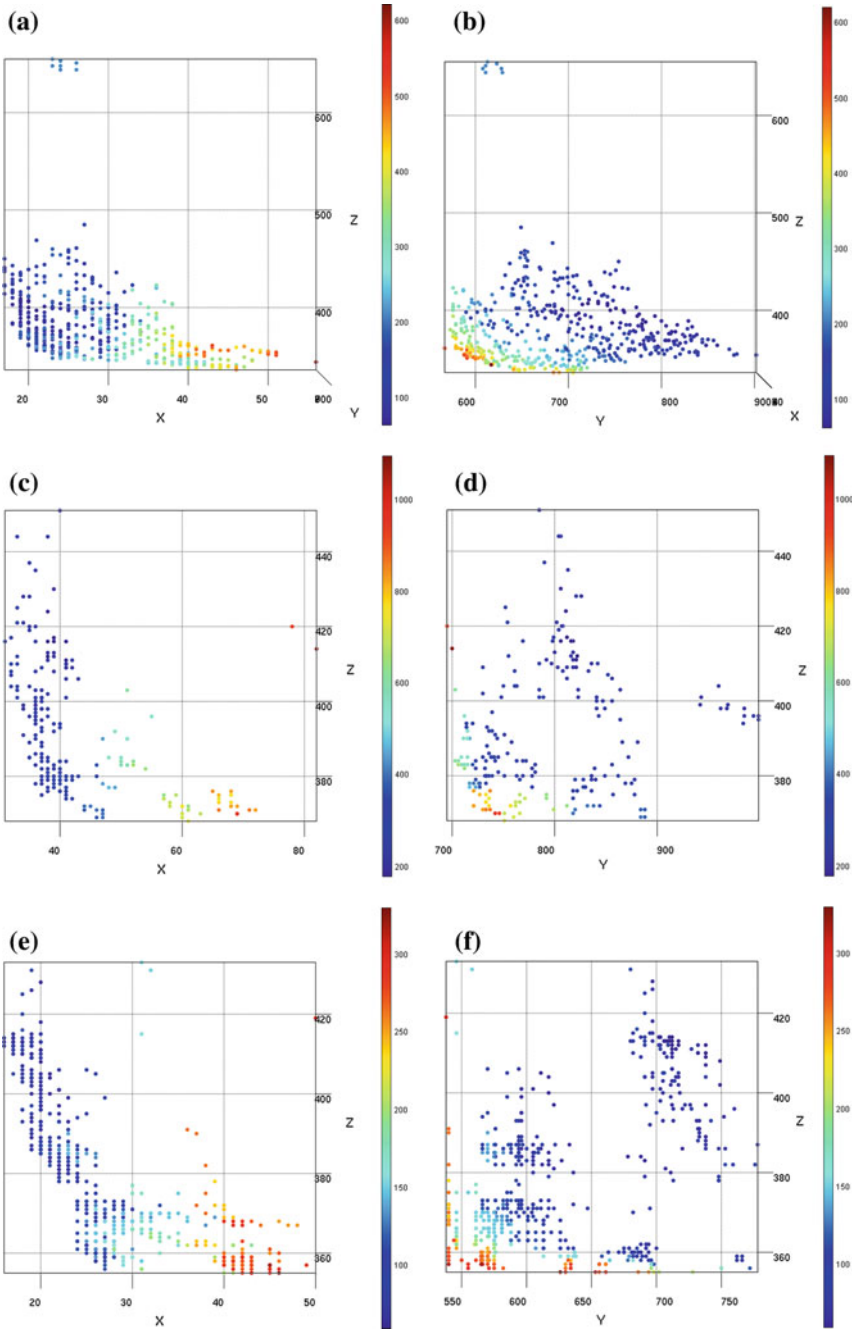

Fig. 4 Pareto front for Task 2 in the XZ and YZ plane: **a** NSGA-II XZ, **b** NSGA-II YZ, **c** SPEA2 XZ, **d** SPEA2 YZ, **e** IBEA XZ, and **f** IBEA YZ

values are much lower at the IBEA algorithm, followed by the NSGA-II and lastly the SPEA2 algorithm.

In Fig. 4 we can see that for the Task 2 the pareto front in regard to the XZ an YZ planes is not pronouncedly divided in two parts as noticed with Task 1. This indicates that the n_{workers} objective does not always "negatively" influences other objectives.

In Table 2 we can see the performance of all algorithms in Task 2, where their pareto min, max, and median values for all objectives are shown. When comparing the objective n_{orders} we can see that the range of values is more or less the same for the IBEA and NSGA-II, while the range is a bit larger for SPEA2; also the median values are lower for the IBEA and NSGA-II. For the objective n_{workers} the IBEA has a little bit smaller and lower range than the NSGA-II and SPEA2, and also the IBEA has the lowest median value. For the objective t_{change} the IBEA and SPEA2 have smaller range of values, but a little bit higher median value than the NSGA-II. For the objective n_{days} the range of values is the smallest and lowest for the IBEA, and also the median value of the IBEA is the lowest one. Similarly as with Task 1, the IBEA was able to find the lowest solution values for objectives except t_{change}, where the NSGA-II was able to find the lowest value. The median value, which shows where the focus of search is, follows the same pattern.

Considering all the information provided by the figures and tables one can conclude that the two tested tasks of presented real-world problem confirm the superiority of the IBEA over the NSGA-II and SPEA2, as shown by using benchmark functions in [16]. So, in this case test benchmark functions proved as a good indication, which is the most suitable algorithm for the job.

7 Conclusion

The multi-objective optimization approach has become important part in the industrial production scheduling, where its goal is to find a production schedule that satisfies different, usually contradictory, production and business constraints. We implemented memetic versions of the NSGA-II, SPEA2 and IBEA multi-objective algorithms, with different approaches to problem solving. These memetic algorithms were applied to real order-lists from a production company. We have shown that for the tested real-world problem the IBEA confirmed its superiority over the NSGA-II and SPEA2, as already indicated by the synthetic test benchmark functions.

Acknowledgements The authors acknowledge the financial support from the Slovenian Research Agency (research core funding No. P2-0098). This work is also part of a project SYNERGY that has received funding from the European Unions Horizon 2020 research and innovation programme under grant agreement No 692286. This work is also part of a project MANTIS that has received funding from the ECSEL Joint Undertaking under grant agreement No 662189. This Joint Undertaking receives support from the European Unions Horizon 2020 research and innovation programme and Spain, Finland, Denmark, Belgium, Netherlands, Portugal, Italy, Austria, United Kingdom, Hungary, Slovenia, Germany.

References

1. Bäck, T., Fogel, D., Michalewicz, Z.: Evolutionary Computation 1: Basic Algorithms and Operators, 2nd edn. Taylor & Francis Group, Heidelberg (2000)
2. Chan, F.T., Au, K., Chan, P.: A decision support system for production scheduling in an ion plating cell. Expert Syst. Appl. **30**(4), 727–738 (2006)
3. Czogalla, J., Fink, A.: On the effectiveness of particle swarm optimization and variable neighborhood descent for the continuous flow-shop scheduling problem. In: Xhafa, F., Abraham, A. (eds) Metaheuristics for Scheduling in Industrial and Manufacturing Applications, volume 128 of Studies in Computational Intelligence, pp. 61–89. Springer, Berlin, Heidelberg (2008)
4. Deb, K.: Multi-Objective Optimization using Evolutionary Algorithms. Wiley-Interscience Series in Systems and Optimization. Wiley, Chichester (2001)
5. Deb, K., Agrawal, S., Pratap, A., Meyarivan, T.: A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In: Schoenauer, M., Deb, K., Rudolph, G., Yao, X., Lutton, E., Merelo, J., Schwefel, H.-P. (eds.) Parallel Problem Solving from Nature PPSN VI, volume 1917 of Lecture Notes in Computer Science, pp. 849–858. Springer, Berlin. Heidelberg (2000)
6. Durillo, J.J., Nebro, A.J.: jmetal: a java framework for multi-objective optimization. Adv. Eng. Softw. **42**(10), 760–771 (2011)
7. Ishibuchi, H., Tsukamoto, N., Nojima, Y.: Evolutionary many-objective optimization: a short review. IEEE Congr. Evol. Comput. CEC **2008**, 2424–2431 (2008)
8. Kamrul Hasan, S.M., Sarker, R., Essam, D., Cornforth, D.: Memetic algorithms for solving job-shop scheduling problems. Memet. Comput. **1**(1), 69–83 (2009)
9. Korošec, P., Bole, U., Papa, G.: A multi-objective approach to the application of real-world production scheduling. Expert Syst. Appl. **40**(15), 5839–5853 (2013)
10. Michalewicz, Z., Fogel, D.: How to Solve It: Modern Heuristics, 2nd edn. Springer, Berlin, Heidelberg (2004)
11. Nareyek, A. (ed.): Local Search for Planning and Scheduling, ECAI 2000, Berlin, Germany, volume 2148 of Lecture Notes in Computer Science. Springer (2000)
12. Papa, G.: Parameter-less algorithm for evolutionary-based optimization. Comput. Optim. Appl. **56**(1), 209–229 (2013)
13. Papa, G., Vukašinović, V., Korošec, P.: Guided restarting local search for production planning. Eng. Appl. Artif. Intell. **25**(2), 242–253 (2012)
14. Wagner, T., Beume, N., Naujoks, B.: Pareto-, aggregation-, and indicator-based methods in many-objective optimization. In: Obayashi, S., Deb, K., Poloni, C., Hiroyasu, T., Murata, T. (eds.) Evolutionary Multi-Criterion Optimization, volume 4403 of Lecture Notes in Computer Science, pp. 742–756. Springer, Berlin, Heidelberg (2007)
15. Zhang, R., Wu, C.: A hybrid local search algorithm for scheduling real-world job shops with batch-wise pending due dates. Eng. Appl. Artif. Intell. **25**(2), 209–221 (2012)
16. Zitzler, E., Künzli, S.: Indicator-based selection in multiobjective search. In: Proceedings of the 8th International Conference on Parallel Problem Solving from Nature (PPSN VIII), pp. 832–842. Springer (2004)
17. Zitzler, E., Laumanns, M., and Thiele, L.: SPEA2: Improving the strength pareto evolutionary algorithm for multiobjective optimization. In: Giannakoglou, K.C., Tsahalis, D.T., Périaux, J., Papailiou, K.D., Fogarty, T. (eds.) Evolutionary Methods for Design Optimization and Control with Applications to Industrial Problems, pp. 95–100 (2001)
18. Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C.M., da Fonseca, Grunert: V.: performance assessment of multiobjective optimizers: an analysis and review. IEEE Trans. Evol. Comput. **7**(2), 117–132 (2003)