Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Computer Science 18 (2013) 936 – 945

Procedia Computer Science

International Conference on Computational Science, ICCS 2013

Optimizing bicriteria flow shop scheduling problem by simulated annealing algorithm

Pempera Jarosław[a], Smutnicki Czesław[a], Żelazny Dominik[a,*]

[a]*Institute of Computer Engineering, Control and Robotics, Wrocław University of Technology, Janiszewskiego 11-17, 50-372 Wrocław, Poland*

Abstract

We consider the flow shop scheduling problem with minimizing two criteria simultaneously: the total completion time (makespan) and the sum of tardiness of jobs. The problem is strongly NP-hard, since for each separate criteria the problem is strongly NP-hard. There is a number of heuristic algorithms to solve the flow shop problem with various single objectives, but usage of those heuristics to multi-criteria flow shop problems is rather limited. In this paper we propose a new idea of the use of simulated annealing method to solve certain multi-criteria problem. Especially, we define a new acceptance rules and the mechanism of moving the search in different regions of solution space by using so called *drift*. To illustrate quality of the proposed approach, we present results of the computational experiment provided on well known benchmarks.

Keywords: flowshop; simulated annealing; bicriteria optimization;

1. Introduction

In the quick changing market the production scheduling plays a key role in manufacturing systems of an enterprise wanting to maintain competitive position. Due to that competition, developing effective and efficient advanced scheduling technologies is extremely important. The so-called flow shop scheduling problem represents a class of widely studied cases based on ideas derived from production engineering, which currently modelled a lot of manufacturing systems, assembly lines, information service facilities [1], and has earned a reputation of being NP-hard to solve [2]. Most of the currently used single objective problems are easily adaptable to real world applications, but modern production scheduling problems need more complex models and thus single-criteria models and algorithms seem to be poor.

Since pioneer works of J.R. Jackson and E.R. Smith in the late fifties of the previous century, the Permutation Flow Shop Scheduling Problem (PFSSP) has received considerable theoretical, computational, and empirical research work. PFSSP is the notorious case in the scheduling theory, commonly considered as a practical scheduling problem with still relatively simple mathematical model. Because of its complexity, branch and bound techniques and classical mathematical programming [3], which provide exact solution, are applicable to only small-scale instances. It led to a lot of various approximate solution methods being proposed, including constructive heuristics, improvement metaheuristics, and memetic algorithms (hybrid). Multi-objective PFSSP

*Corresponding author: Dominik Żelazny.
E-mail address: dominik.zelazny@pwr.wroc.pl.

Selection and peer review under responsibility of the organizers of the 2013 International Conference on Computational Science
doi:10.1016/j.procs.2013.05.259

constitutes natural evolution of models and solution methods, oriented on practice. Actually, scheduling decisions usually have to take into account several economic indexes simultaneously. For more than a decade, a number of multi-objective algorithms have been suggested. Primarily because of their ability to find multiple Pareto-optimal solutions (an aproximation of the Pareto frontier) in single run. Since it is not possible to have a single solution simultaneously optimizing all objectives, algorithms that give solutions lying on or near the Pareto-optimal front are of great practical value to companies and factories.

2. Multi-objective optimization in literature

The literature on multi-objective optimization is in abundance, albeit the multi-criteria PFSP has not received such interest. Especially in relation to the number of works on flow shop problems with single criterion. Most of the multi-criteria PFSP papers are either based on branch and bound methods or evolutionary algorithms (EA), while only some consist of local search methods like tabu search (TS) or simulated annealing (SA).

2.1. General multi-objective algorithms

Among the existing elitist multi-objective evolutionary algorithms (MOEAs), Zitzler and Thiele's [4] strength Pareto EA (SPEA), Knowles and Corne's Pareto-archived evolution strategy (PAES) [5], and Rudolph's [6] elitist GA are well known. Strength Pareto EA (SPEA) is an elitist multi-criterion EA with the concept of non-domination. Zitzler and Thiele [4] suggested maintaining an external population at every generation storing all discovered non-dominated solutions. It participates in genetic operations. All non-dominated solutions are assigned a fitness based on the number of solutions they dominate, while dominated solutions are assigned a fitness worse than the worst fitness of any non-dominated solution, so that the search is directed towards the non-dominated solutions. A clustering technique is used to ensure diversity among non-dominated solutions.

In Pareto-archived ES (PAES), the child is compared with respect to the parent. If the child dominates the parent solution, then the parent is discarded and the child takes its place as the next parent. If the child is dominated by the parent, then the child is discarded and new child solution is generated. On the other hand, if the child and the parent do not dominate each other, the child is compared with the archive to check if it dominates any member of the archive of non-dominated solutions. If so, the child is accepted as the new parent and the dominated solutions are eliminated from the archive. Else, both parent and child are checked for their nearness with the solutions of the archive and the one residing in a least crowded region in the parameter space is accepted as the parent and added to the archive.

Rudolph [6] suggested, a simple elitist multi-objective EA based on a systematic comparison of individuals from parent and offspring populations. The non-dominated solutions of both offspring and parent populations are compared, to form new set of non-dominated solutions, which becomes the parent population in the next iteration. If the size of this set is lower then the desired population size, then other solutions from the offspring population are included. Unfortunately this algorithm lacks in the task of maintaining diversity of Pareto-optimal solutions.

In [7] Deb suggested an Elitist Non-dominated Sorting Genetic Algorithm (NSGA-II). It was based on the non-dominated sorting GA (NSGA), criticised for high computational complexity of non-dominated sorting, lack of elitism and need for specifying the sharing parameter, it modified its approach to alleviate those difficulties. Applying fast non-dominated sorting, density estimation and crowded comparison operator allowed it to lessen the computational complexity and guide the selection process of the algorithm towards a uniformly spread out Pareto-optimal front.

Moreover, recent results show clearly that elitism can speed up the performance of the genetic algorithms significantly. It also helps to prevent the loss of good solutions, once they have been found.

2.2. Multi-criteria optimization in flow shop problems

Most commonly used multi-criteria algorithms for flow shop problems use Pareto efficiency evaluation, which is considered as one of the best approaches to the appraisal of solutions. Although there were some attempts at non-Pareto algorithms as well.

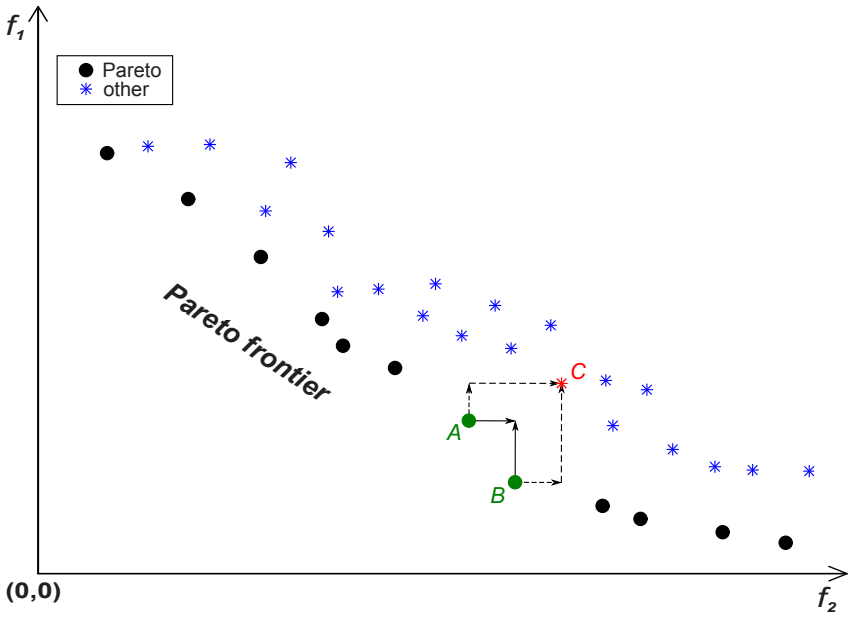

Fig. 1. Sample of Pareto frontier and dominated points

2.2.1. Pareto efficiency

The solution to a multi-objective problem is the set of non-dominated solutions called the Pareto frontier, where dominance is defined as follows. A solution $y = (y_1, y_2, ..., y_n)$ dominates (denoted $\prec$) a solution $z = (z_1, z_2, ..., z_n)$ if and only if $\forall_i \in \{1...n\}$, $y_i \leq z_i$ and $\exists_i \in \{1...n\}$, $y_i < z_i$. Fig. 1 clearly shows dominance of points A and B over point C and lack of such between aforementioned points A and B, of which both are non-dominated and as such included in the approximation of the Pareto frontier.

2.2.2. Literature on MO PFSP algorithms

The Multi-Objective Genetic Algorithm (MOGA) of Murata et al. [8], being part of evolutionary algorithms, was developed to solve multi-objective flow shop problem. Other than a modified selection operator, this algorithm was a simple genetic approach to scheduling. Selection is interrelated with a set of weights assigned to the objectives, which allowed to distribute the search towards different criteria directions. Elitist preservation method was also incorporated, so that several solutions from the actual Pareto frontier were copied to the next generation. The MOGA was furthermore enhanced by Murata et al. [9], by changing the way of weight distribution between objectives. Using a cellular structure permitted a better weight selection, which in turn led to finding a finer approximation of Pareto frontier. New algorithm was called CMOGA.

A representative of the local search method could be found in the work of Chakravarthy and Rajendran [10]. Their goal was to minimize the weighted sum of two objectives using a simple SA algorithm. Initial solution is selected from the following methods: a) Earlies Due Date (EDD), Least Static Slack (LSS) and NEH heuristic [11], while generating a neighbourhood was performed by the adjacent interchange scheme (AIS). Since it uses weighted objectives, this algorithm does not belong to the set of Pareto approach algorithms.

Inspired by the Pareto-archived ES algorithm, Suresh and Mohanasundaram proposed a Pareto Archived Simulated Annealing (PASA) [12], in which new perturbation was suggested. Mechanism called Segment Random Insertion was used to generate the neighbourhood of a given sequence. In order to retain non-dominated solutions, an external archive is used. Initial solution is randomly generated, while new current solution is selected by a scaled weighted sum of the objective values.

Variation of genetic algorithm, with an initialization procedure which inserts four good solutions into initial random population, was proposed by Pasupathy et al. in [13]. It used an external population, for non-dominated solutions. Evolution strategy is similar to the one used in NSGA-II, while crowding distance procedure is used

as a secondary population selector. Improving the quality of Pareto frontier is based on two different local search procedures, applied to the external population afterwards.

3. Problem description

Let us consider a manufacturing system with a structure consisting of m machine given by the set $M = \{1, ..., m\}$. Consider a set $J = \{1, 2, ..., n\}$ of jobs to be processed on machines. Each job has to be processed on a machine $1, 2, ..., m$ in that order. Job j, $j \in J$, consists of a sequence of m operations $O_{j,1}, O_{j,2}, ..., O_{j,m}$. Operation $O_{j,k}$ corresponds to the processing of job j on a machine k during an uninterrupted processing time $p_{j,k} \geq 0$. In the permutation flow shop problem the job sequence must be the same on all machines. Each machine can execute at most one job at a time and each job can be processed on at most one machine at a time. Each job $j \in J$ should be delivered before its due date $d_j \geq 0$.

The schedule of jobs (solution of the problem) can be described by starting $S_{j,k}$ and completion times $C_{j,k}$ of operations $j \in J$, $k \in M$, satisfying the above mentioned constraints. The operation $O_{j,k}$ starts in $S_{j,k}$ and completes in $C_{j,k}$. The job j, $j \in J$ is delivered by the production system in the time moment $C_{j,m}$. We consider two objective functions: the total completion time $C_{\max}$ and the total tardiness T_{tot}. For a given schedule described by $C_{j,k}$ the objective values can be calculated with the following expressions:

$$C_{\max} = \max_{j \in J} C_{j,m} \tag{1}$$

and

$$T_{tot} = \sum_{j \in J} T_j, \tag{2}$$

where $T_j = \max\{0, C_{j,m} - d_j\}$ is a tardiness of job $j \in J$.

In the paper we refer to another (equivalent) characterization of the solution which uses loading sequence instead of the schedule. Let a permutation π of n jobs determine the processing order on all machines. The completion times can be calculated with the well known recursive expression:

$$C_{\pi(j),k} = \max\{C_{\pi(j),k-1}, C_{\pi(j-1),k}\} + p_{\pi(j),k}, \tag{3}$$

where $C_{\pi(j),0} = 0$, $\forall j \in J$, $C_{0,k} = 0$, $\forall k \in M$, $\pi(j)$ denotes the job in the j-th position of the sequence. The completion times obtained from (3) are as small as possible. Finally, due to regularity of both objectives, for a given processing order described by π, the minimal value of objectives can be calculated with the following expressions:

$$C_{\max}(\pi) = \max_{1 \leq s \leq n} C_{\pi(s),m} = C_{\pi(n),m}, \tag{4}$$

and

$$T_{tot}(\pi) = \sum_{s=1}^{n} T_{\pi(s)}, \tag{5}$$

where $T_{\pi(s)} = \max\{0, C_{\pi(s),m} - d_{\pi(s)}\}$ is a tardiness of job $\pi(s) \in J$.

For the multi-objective function $F(\pi) = (f_1(\pi), ..., f_q(\pi))$, $q = 2$, $f_1(\pi) = C_{\max}(\pi)$, $f_2(\pi) = T_{tot}(\pi)$, we want to find Pareto optimal set of solution $\Pi^* \subset \Pi$. The set Π^* consists of non-dominated solutions of set Π. A solution α dominates solution π if and only if

$$f_k(\alpha) \leq f_k(\pi) \quad \forall k \in \{1, ..., q\} \tag{6}$$

$$f_k(\alpha) < f_k(\pi) \quad \exists k \in \{1, ..., q\} \tag{7}$$

4. Simulated annealing

Simulated annealing (SA) is a local search method proposed originally by Kirkpatrick et al. [14] which uses analogy to a thermodynamic cooling process to avoid local minima and escape from them. States of a solid are viewed as being analogous to solutions, whereas the energy of the solid relates to an objective function. Despite the analogy to complicated physical process, implementation of algorithms based on this method is relatively simple. Moreover, the high efficiency of this method of algorithms construction is confirmed by the numerous propositions of SA-based algorithms for many hard problems of scheduling optimization: the resource-constrained project scheduling problem [15], the job shop scheduling problem with total weighted tardiness objective [16], the multi-mode resource-constrained project scheduling with positive discounted cash flows and different payment models [17]. For these reasons, this method was highly popular among researchers for years.

Primal idea of SA was formulated for the case of optimization of a single objective function f(x). In each iteration, for a current solution x, neighbour x' is selected from the neighbourhood of x, denoted as $N(x)$. The neighbour is chosen in an ordered way or randomly, assuming usually uniform distribution of probability. The objective difference $\Delta = f(x') - f(x)$ is evaluated. The solution x' replaces x whenever $\Delta \leq 0$. Otherwise, x' is accepted as the new solution for the next iteration with probability $p = exp(-\Delta/T)$, where T is parameter called temperature. Starting from the initial temperature T_0, the temperature is reduced slowly with each iteration using cooling scheme. The most commonly used scheme is geometric, in which the temperature during $s - th$ iteration is $T_s = \lambda T_0$, where λ is a parameter.

Designing an algorithm, based on the SA method, for a particular problem, starts from defining: a solution representation of the problem and neighbourhood generation, a method for the objective function value calculation, a cooling scheme and including stopping criteria.

5. The proposed multi-objective SA approach

In recent years, SA has been developed to cover multiple-criteria case. The most known of all simulated annealing algorithms, commonly used by numerous researchers, is the algorithm proposed by Varadharajan and Rajendran [18]. This algorithm was originally proposed for scheduling in flow shops to minimize the makespan and total flow time of jobs. The authors used multi-run version of SA algorithm and introduce direction vector to intensify search of different regions of Pareto frontier. It's worth to mention, that computation results clearly show that algorithms based on SA method outperform algorithms based on genetic search method [8] and [19]. The multi-objective version of SA were successfully used for multi-objective optimization in real-life problem. In their work, Loukil, Teghem and Fortemps [20] consider the scheduling in a Tunisian company.

At the beginning of the description of proposed algorithm we will return to general discussion about SA method. In SA-based algorithm, two values influence probability of acceptance worse solution x', $f(x') > f(x)$, for the next step of the algorithm: the temperature T and the objective difference $\Delta = f(x') - f(x)$. Higher values of temperature T give a higher probability of acceptance. The influence of quality of solution (measured by Δ) on acceptance probability is opposite.

In a typical run of well-designed SA algorithm, we can distinguish three phases: initial, major and final. In the initial phase, the current solution is significantly improved. This is accomplished through the rejection of solutions considerably worse than the current solution. In the major phase, the good solution found in the first phase is improved by search, which allows acceptancing solutions only a little worse than the current solution. In this phase, the region consisting of good quality solutions is searched. In the final phase the acceptance probability is so small that in practice only solutions not worse then current one are accepted. In the last phase of algorithm the sequence of different solutions with the same objective values is generated. This effect is called a drift. The effect of drift is a very advantageous feature of the SA algorithm, because it allows searching a flat space of solutions.

Application of the SA method for the construction of algorithms for multi-criteria optimization problems requires answers to many questions: how to determine whether a solution x' is worse than the x, and if so, how to determine a set of solutions (Pareto set)?

The solutions x and x' can be in three mutual relations: (i) x' dominates x, (ii) x dominates x', (iii) x' and x are incomparable. In the case (i) it is obvious that the solution x' is better than x and it replaces it in the next iteration. Similarly, in the case (ii) the solution x' is worse than x, so we must determine by how much. Let

the $\Delta = \sqrt{\sum_{i=1}^{q}(f_i(x') - f_i(x))^2}$ be a scalar measure of difference between objective function values for solution x' and and solution x. The value Δ is no less than $\max_{1\leq i\leq q}\{f_i(x') - f_i(x)\}$ and each difference $f_i(x') - f_i(x)$, $i = 1, ..., q$ influence this value i.e. increase it. Finally, in case (*iii*) for at least one pair of functions $f_k()$ and $f_l()$, $k, l \in \{1, ..., q\}$ occurs $f_k(x') < f_k(x)$ and $f_l(x') > f_l(x)$. Thus, at least one function is improved and at least one function is deteriorated. Similarly to the drift effect, the acceptance of such type of solutions moves the search process in other region of Pareto frontier, i.e region with smaller value of some functions and greater of others. Note, that solutions of (*i*) and (*iii*) type are generated very rarely.

Figure 2 shows a typical run of proposed SA algorithm. The plot consists of three types of points. Empty circles denote points of function $minT(x) = \min\{T_{tot}(\pi) : C_{\max}(\pi) = x, \pi \in \Pi\}$, while the remaining – denote points generated by SA algorithm. Non-dominated solutions generated by SA algorithm were drawn as filled circles. Points of function $minT(x)$ were generated by the random walk algorithm, which generates 10 millions random solutions while SA only generates 10 thousand. It is easy to see the overall relationship between both criteria. The minimal total tardiness for the given makespan increases with increasing makespan. It does not apply to a small range of functions which are Pareto optimal points. All solutions generated by SA algorithm are situated in near distance of the Pareto frontier.

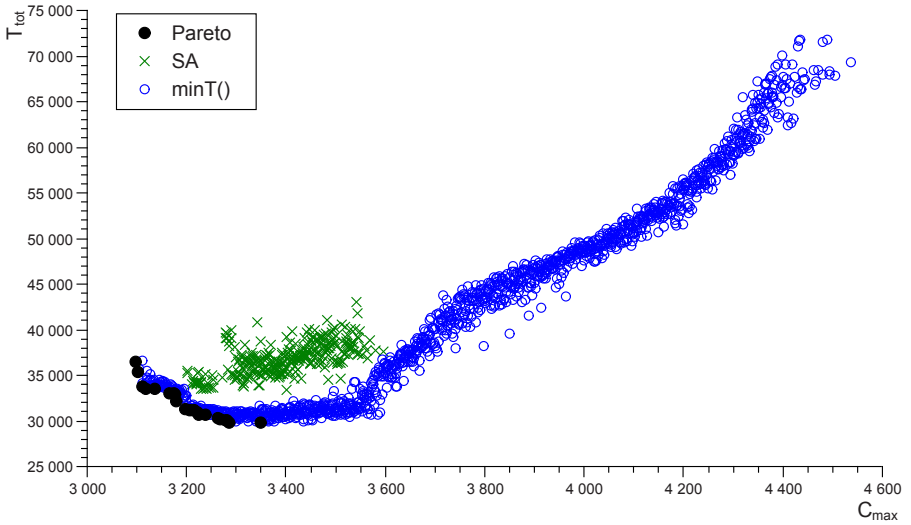

Fig. 2. Typical run of the proposed algorithm

6. Computational evaluation

The aim of the experiment was to compare the effectiveness of a proposed algorithm with the benchmarks taken from literature. The algorithms were tested on 110 benchmark instances provided by Taillard [21] for the flow-shop problem and modified by Ruiz [22] for total tardiness criterion. The benchmark set contains 11 groups of 'hard' instances of different sizes $n \times m$: 20×5, 20×10, 20×20, 50×5, 50×10, 50×20, 100×5, 100×20, 200×10, 200×20.

We implemented multi-run version of proposed algorithm in C++ language. Each run, excluding the first, begins from randomly selected non-dominated solution. All found non-dominated solutions were archived in the list, which was updated each time a new solution was generated. The initial temperature for each run was 100, while the finish temperature was 1. In a single run SA algorithm performed 10,000 iteration (the parameter λ was accordingly calculated). The algorithm was run 320 times, so for each instance 3,200,000 solutions were generated.

The implementation was executed and tested on the Compaq 8510w Mobile Workstation PC with Intel Core 2 Duo 2.60 GHz processor. The computation time of program varied from 60 s for instances from 20×5 group to

1,200 s for the biggest instances i.e. for 20×5 group. Note, that the computation times are comparable with those of heuristics tested in [22].

Comparing multi-objective algorithms is not as apparent as comparing the ones with single criterion, where lower/higher (minimization/maximization) value of solution translates directly to a better solution. Considering different methods (scalar, Pareto and others) of solution evaluation, there is no single way to evaluate which set of non-dominated solutions is clearly better than the other. Our approach is based on a dominance relation, so a returned result for each instance a set of non-dominated solutions, called approximation of Pareto frontier. There is a number of methods developed and used to compare such non-dominated sets.

6.1. Quality indicators and reference data

In this paper we decided to use two ways of comparing our results with data provided by Ruiz et al. in [22]. Their benchmarks were prepared using 16 different algorithms and running each 10 times, out of all the obtained results approximated Pareto frontiers were published.

6.1.1. Number of Pareto efficient solutions

In our earlier work [23] we devised a method of comparison, which used a percentage of non-dominated solutions in the aggregation of compared sets as an indicator. Solutions from all the algorithms were flagged and aggregated into a single set, which was then purged of dominated solutions. A number of solutions in this global Pareto-efficient set was computed for each algorithm and those numbers were compared to evaluate solution sets.

6.1.2. Hypervolume Indicator

Knowles et al. [24] provided a few necessary tools for a better evaluation and comparison of multi-objective algorithms. They proposed, among others, a hypervolume indicator I_H to measure quality of the Pareto frontier approximations. Hypervolume indicator measures the area covered by the approximated Pareto frontiers for each of the algorithms. In order to bound this area, a reference point is used. A greater value of I_H indicates both a better convergence to as well as a good coverage of the optimal Pareto front. In our case, reference points were calculated as follows. For each of the criteria used we took worst value from both (ours and comparative) non-dominated sets, multiplied it by 1.2 and assigned as reference points value of that criteria. The very same method was proposed in [22].

Table 1. Comparision summary of obtained results.

Instance size	Reference PS	Our PS	Aggregated PS	New PS	Dominated ref. PS	Coverage
20×5	227	181	294	67	1	0.986
20×10	390	258	419	29	0	0.962
20×20	318	200	329	11	0	0.942
50×5	182	198	184	4	3	0.960
50×10	406	221	406	5	10	0.868
50×20	745	303	754	9	0	0.844
100×5	219	296	306	102	54	0.996
100×10	365	293	380	15	0	0.901
100×20	509	362	558	49	0	0.846
200×10	388	197	388	0	0	0.878
200×20	589	189	589	0	0	0.827
$\sum$	4338	2698	4607	291	68	0.910 (avg)

6.2. Results

There are 110 instances divided into 11 instance sizes, thus computation results were combined into instance size categories. By PS we refer to the Pareto Solutions number, while Coverage is quotient of our Pareto frontiers I_H and reference frontiers I_H.

As can be observed in Tab. 1, our algorithm was able to find 291 new non-dominated and dominate 68 of reference solutions in total. Detailed analysis of obtained results shows that for 60 instances our coverage of reference I_H was no less than 90% and for 6 instances we were able to find sets of equal or better coverage. We observed, that problems of instance sizes 100×50, 20×5 and 100×20 show the most promising results. For detailed analysis of our research we recommend A.2 and A.3.

7. Conclusion

In comparison to the benchmarks, provided by Ruiz et al.[22], we can conclude that our approach was successful. Albeit it didn't entirely dominate those comparative results, it did find a new segment of Pareto frontier approximation. We used SA algorithm, due to it's high performance in discrete optimization noted by many researchers. We proposed a function to measure scalar difference between two multi-criteria solutions, which allowed the use of acceptance function for such. Also, we develop a new navigation method of search method follow the Pareto frontier. It is based on a drift effect (well known for single criteria), which was described in relation to multi-objective problems in section 5. In case of multi-criteria scheduling, guiding the search process in direction of one criteria allows us to further improve results in that part of Pareto frontier, while maintaining good results in overall search for non-dominated solutions. This feature could be of further use in our research, especially if multiple-criteria decision-making (MCDM) or multiple-criteria decision analysis (MCDA) were to be included.

References

[1] M. Pinedo, Scheduling: Theory, Algorithms and Systems, 2nd ed, Prentice-Hall, 2002.
[2] M. Garey, D. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W. H. Freeman, 1979.
[3] B. Lageweg, J. Lenstra, A. R. Kan, A general bounding scheme for the permutation flowshop problem, Operations Research 26 (1) (1978) 53–67.
[4] E. Zitzler, L. Thiele, Multiobjective optimization using evolutionary algorithms - a comparative case study and the strength pareto approach, Evolutionary Computation, IEEE Transactions 3 (4) (1999) 257–271.
[5] J. Knowles, D. Corne, The pareto archived evolution strategy: A new baseline algorithm for multiobjective optimisation (1999) 98–105.
[6] G. Rudolph, Evolutionary search under partially ordered fitness sets, Tech. rep. (2001).
[7] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multi-objective genetic algorithm: Nsga-ii, IEEE Transaction on Evolutionary Computation 6 (2) (2002) 182–197.
[8] T. Murata, H. Ishibuchi, H. Tanaka, Multi-objective genetic algorithm and its applications to flowshop scheduling, Computers and Industrial Engineering 30 (4) (1996) 957–968.
[9] T. Murata, H. Ishibuchi, M. Gen, Specification of genetic search directions in cellular multiobjective genetic algorithms, Springer (2001) 82–95.
[10] K. Charavarthy, C. Rajendran, A heuristic for scheduling in a flowshop with the bicriteria of makespan and maximum tardiness minimization, Production Planning and Control 10 (7) (1999) 707–714.
[11] M. Nawaz, J. E. E, E, I. Ham, A heuristic algorithm for the m-machine, n-job flowshop sequencing problem, The International Journal of Management Science 11 (1) (1983) 91–95.
[12] R. Suresh, K. Mohanasundaram, Pareto archived simulated annealing for permutation flowshop scheduling with multiple objectives (2004) 712–717.
[13] T. Pasupathy, C. Rajendran, R. Suresh, A multi-objective genetic algorithm for scheduling in flowshops to minimize the makespan and total flowtime of jobs, The International Journal of Advanced Manufacturing Technology 27 (7–8) (2006) 804–815.
[14] S. Kirkpatric, C. Gelatt, M. Vecchi, Optimisation by simulated annealing, Science (220) (1983) 671–680.
[15] K. Bouleimen, H. Lecocq, A new efficient simulated annealing algorithm for the resource-constrained project scheduling problem and its multiple mode version, European Journal of Operational Research (149) (2003) 268–281.
[16] R. Zhang, C. Wu, A simulated annealing algorithm based on block properties for the job shop scheduling problem with total weighted tardiness objective, Computers & Operations Research (38) (2011) 854–867.
[17] M. Mika, G. Waligora, J. Weglarz, Simulated annealing and tabu search for multi-mode resource-constrained project scheduling with positive discounted cash flows and different payment models, European Journal of Operational Research (164) (2005) 639–668.
[18] T. Varadharajan, C. Rajendran, A multi-objective simulated-annealing algorithm for scheduling in flowshops to minimize the makespan and total flowtime of jobs, European Journal of Operational Research (167) (2005) 772–795.
[19] T. Bagchi, Multiobjective Scheduling by Genetic Algorithms, Kluwer Academic Publishers, Boston, 1999.
[20] T. Loukil, J. Teghem, P. Fortemps, A multi-objective production scheduling case study solved by simulated annealing, European Journal of Operational Research (179) (2007) 709–722.
[21] E. Zitzler, L. Thiele, Benchmarks for basic scheduling problems, European Journal of Operational Research 64 (2) (1993) 278–285.
[22] R. Ruiz, G. Minella, M. Ciavotta, A review and evaluation of multi-objective algorithms for the flowshop scheduling problem, Journal on Computing 20 (3) (2008) 451–471.
[23] J. Pempera, C. Smutnicki, D. Żelazny, Multi-objective optimization of production schedules, in: Proc. of the Production Engineering, 2011.
[24] J. Knowles, L. Thiele, E. Zitzler, A tutorial on the performance assessment of stochastic multiobjective optimizers, Tech. rep., ETH Zurich (2006).

Appendix A. All results for PFSP Taillards instances with Due Dates.

All of collected data, that were summarized earlier.

Table A.2. Comparision of instances 001 - 045.

Instance size	Reference PS	Our PS	Aggregated PS	New PS	Dominated ref. PS	Coverage
DD Ta001	8	9	15	7	0	1.00
DD Ta002	13	16	16	3	1	0.99
DD Ta003	17	20	23	6	0	0.99
DD Ta004	55	33	64	9	0	0.97
DD Ta005	24	20	39	15	0	1.00
DD Ta006	18	12	21	3	0	0.98
DD Ta007	22	19	29	7	0	0.99
DD Ta008	21	13	28	7	0	0.99
DD Ta009	22	17	25	3	0	0.97
DD Ta010	27	22	34	7	0	0.98
DD Ta011	26	17	26	0	0	0.96
DD Ta012	40	30	44	4	0	0.98
DD Ta013	47	31	48	1	0	0.96
DD Ta014	42	27	47	5	0	0.97
DD Ta015	26	18	26	0	0	0.94
DD Ta016	67	44	73	6	0	0.96
DD Ta017	23	20	25	2	0	0.98
DD Ta018	20	13	20	0	0	0.94
DD Ta019	46	25	53	7	0	0.96
DD Ta020	53	33	57	4	0	0.97
DD Ta021	20	9	20	0	0	0.92
DD Ta022	40	27	40	0	0	0.96
DD Ta023	13	9	13	0	0	0.93
DD Ta024	3	5	3	0	0	0.88
DD Ta025	55	34	65	10	0	0.98
DD Ta026	68	40	69	1	0	0.96
DD Ta027	25	15	25	0	0	0.94
DD Ta028	54	30	54	0	0	0.96
DD Ta029	30	19	30	0	0	0.93
DD Ta030	10	12	10	0	0	0.96
DD Ta031	12	15	12	0	0	0.97
DD Ta032	17	32	17	0	0	0.95
DD Ta033	15	17	15	1	1	0.95
DD Ta034	31	25	31	0	0	0.95
DD Ta035	8	14	8	0	0	0.97
DD Ta036	18	19	18	0	0	0.95
DD Ta037	16	17	16	1	2	0.98
DD Ta038	17	18	17	0	0	0.94
DD Ta039	32	23	32	0	0	0.96
DD Ta040	16	18	18	2	0	0.98
DD Ta041	37	23	37	0	0	0.88
DD Ta042	37	22	37	0	0	0.92
DD Ta043	41	19	41	0	0	0.83
DD Ta044	31	26	31	2	3	0.88
DD Ta045	29	19	29	0	0	0.85

Table A.3. Comparision of instances 046 - 110.

Instance size	Reference PS	Our PS	Aggregated PS	New PS	Dominated ref. PS	Coverage
DD Ta046	41	25	41	0	0	0.83
DD Ta047	43	17	43	3	7	0.89
DD Ta048	40	24	40	0	0	0.85
DD Ta049	58	27	58	0	0	0.85
DD Ta050	49	19	49	0	0	0.90
DD Ta051	56	39	64	8	0	0.84
DD Ta052	79	33	80	1	0	0.87
DD Ta053	81	26	81	0	0	0.85
DD Ta054	69	37	69	0	0	0.90
DD Ta055	104	38	104	0	0	0.85
DD Ta056	78	24	78	0	0	0.82
DD Ta057	70	22	70	0	0	0.83
DD Ta058	67	22	67	0	0	0.83
DD Ta059	67	24	67	0	0	0.79
DD Ta060	74	38	74	0	0	0.86
DD Ta061	12	15	12	6	9	1.00
DD Ta062	21	25	21	0	0	0.95
DD Ta063	28	38	51	28	5	1.12
DD Ta064	15	16	16	4	3	0.95
DD Ta065	28	33	32	5	11	0.99
DD Ta066	15	30	15	0	0	0.95
DD Ta067	20	17	29	9	13	1.02
DD Ta068	37	46	52	15	7	0.98
DD Ta069	16	36	26	10	0	0.99
DD Ta070	27	40	52	25	6	1.01
DD Ta071	41	30	41	0	0	0.89
DD Ta072	44	37	44	0	0	0.91
DD Ta073	30	34	30	0	0	0.88
DD Ta074	42	21	42	0	0	0.87
DD Ta075	46	29	46	0	0	0.89
DD Ta076	31	39	46	15	0	0.93
DD Ta077	42	33	42	0	0	0.89
DD Ta078	35	22	35	0	0	0.92
DD Ta079	33	23	33	0	0	0.92
DD Ta080	21	25	21	0	0	0.91
DD Ta081	58	36	66	8	0	0.85
DD Ta082	64	43	64	0	0	0.85
DD Ta083	51	38	61	10	0	0.89
DD Ta084	45	46	62	17	0	0.84
DD Ta085	44	30	54	10	0	0.84
DD Ta086	44	32	44	0	0	0.84
DD Ta087	67	25	67	0	0	0.86
DD Ta088	35	31	35	0	0	0.82
DD Ta089	56	31	60	4	0	0.85
DD Ta090	45	50	45	0	0	0.82
DD Ta091	28	19	28	0	0	0.90
DD Ta092	51	23	51	0	0	0.87
DD Ta093	22	23	22	0	0	0.92
DD Ta094	18	21	18	0	0	0.91
DD Ta095	54	25	54	0	0	0.87
DD Ta096	29	12	29	0	0	0.87
DD Ta097	64	21	64	0	0	0.85
DD Ta098	27	13	27	0	0	0.87
DD Ta099	42	21	42	0	0	0.84
DD Ta100	53	19	53	0	0	0.88
DD Ta101	59	16	59	0	0	0.85
DD Ta102	72	17	72	0	0	0.81
DD Ta103	62	25	62	0	0	0.82
DD Ta104	64	22	64	0	0	0.83
DD Ta105	57	17	57	0	0	0.84
DD Ta106	44	21	44	0	0	0.80
DD Ta107	51	16	51	0	0	0.82
DD Ta108	53	17	53	0	0	0.84
DD Ta109	64	23	64	0	0	0.82
DD Ta110	63	15	63	0	0	0.84