An International Journal of Optimization
and Control: Theories & Applications
Vol.5, No.1, pp.1-11 (2015) © IJOCTA
ISSN:2146-0957 eISSN:2146-5703
DOI:10.11121/ijocta.01.2015.00232
http://www.ijocta.com

Solving multi-objective job shop problem using nature-based algorithms: new Pareto approximation features

Jarosław Rudy[a] and Dominik Żelazny[a]

[a]Wrocław University of Technology,
Department of Automatic Control and Mechatronics, Poland
Email: {jaroslaw.rudy,dominik.zelazny}@pwr.edu.pl

(Received August 8, 2014; in final form October 20, 2014)

Abstract. In this paper the job shop scheduling problem (JSP) with minimizing two criteria simultaneously is considered. JSP is frequently used model in real world applications of combinatorial optimization. Multi-objective job shop problems (MOJSP) were rarely studied. We implement and compare two multi-agent nature-based methods, namely ant colony optimization (ACO) and genetic algorithm (GA) for MOJSP. Both of those methods employ certain technique, taken from the multi-criteria decision analysis in order to establish ranking of solutions. ACO and GA differ in a method of keeping information about previously found solutions and their quality, which affects the course of the search. In result, new features of Pareto approximations provided by said algorithms are observed: aside from the slight superiority of the ACO method the Pareto frontier approximations provided by both methods are disjoint sets. Thus, both methods can be used to search mutually exclusive areas of the Pareto frontier.

Keywords: Multi-objective optimization; job shop scheduling; multi-criteria decision analysis; nature inspired.
AMS Classification: 90B35, 68M20, 90C27, 90C29.

1. Introduction

Companies are always interested in maintaining competitive position in fast-changing market, this is usually done by optimizing the business and production processes. Due to that competitiveness, access to effective and fast optimization methods is extremely important. This phenomenon results in constant development of new approaches to optimization of various practical problems. The so-called job shop scheduling problem (JSP) represents a class of widely studied cases based on ideas derived from production engineering and has been classified as a NP-hard problem [5]. Most of the currently used single objective models are easily adaptable to real world applications, but modern production scheduling problems need further advancements.

Since its first formulation, JSP has received considerable theoretical, computational, and empirical research work. Thus, JSP is often studied case in the scheduling theory. Due to its complexity, branch and bound techniques and classical mathematical programming providing exact solutions, are applicable only to small-scale instances. Hence, a lot of various approximate solution methods were proposed, including constructive heuristics, improvement meta-heuristics, and hybrid algorithms. Multi-objective scheduling is the result of natural evolution of models and solution methods, oriented on practice, since scheduling decisions usually have to take into account several economic indexes simultaneously. In the

Corresponding Author. Email: dominik.zelazny@pwr.edu.pl.

last few decades, a number of multi-objective evolutionary algorithms have been suggested, primarily due to their ability to find approximation of the Pareto front in single run. Other population-based approaches share similar benefits of evolutionary techniques. In most practical cases, it is not possible to have a single solution simultaneously optimizing all objectives, therefore algorithms that provide solutions lying on or near the Pareto efficient front are of great practical value.

The aim of this paper is to provide new insight into the possible solutions to the MOJSP and the quality of those solution. Thus, we use metaheuristic algorithms created specifically for applying the to the MOJSP and analyze the obtained approximations of Pareto fronts in order to establish new properties. Moreover, according to our research, the literature on MOJSP lacks multi-criteria benchmarks usable in research, thus we also aim to provide such benchmarks obtained from our own studies.

The remainder of the paper is organized as follows. In Section 2 we present an overview of the literature on the JSP problem, including the multi-objective approach. Section 3 contains problem description and Section 4 briefly presents approaches to evaluation in multi-criteria optimization. Sections 5 and 6 contain the description of our approach, including solution representation and metaheuristic algorithms. Section 7 contains the computer experiment results. Finally, Sections 8 and 9 contain the conclusions and acknowledgements.

2. Literature on the JSP

JSP is a NP-hard discrete optimization problem. Exact algorithms, like branch and bound methods, have been proposed for instances of small sizes and allowed to find optimal solutions. However, larger instances found in real-world applications were a contribution to the creation of approximate methods. Those methods allowed to find near optimal solutions in reasonable computation time, and many heuristics and metaheuristics were proposed and developed for different objective functions. Designing any algorithm for JSP requires to designate solution representation to be used and, in most cases, decoding scheme. Many research approaches concerning the JSP exist. Most of them are aimed at the single criterion version of the problem, while our interest lies in the multi- objective JSP. Therefore, we will focus on the literature for the multi-object version of the problem, while limiting the literature on single criterion case to the required minimum.

There has been a number of papers proposing metaheuristic techniques for solving JSP. Despite being NP-hard optimization problem, many researchers made efforts to develop efficient algorithms for solving it. Some of the best known proposed approaches include tabu search by Nowicki and Smutnicki [13]. Moreover, methods like simulated annealing (SA), evolutionary algorithms (EA), ant colony optimization (ACO) and particle swarm optimization (PSO) were proposed for JSP. In recent years, researchers applied parallel methods for solving JSP. Such parallel-oriented works included greedy randomized adaptive search procedures, SA, EA methods and various local search (LS) methods. Bożejko *et al.* [1] introduced parallel SA, which employed properties of the problem associated with the *block theory*. Additionally, the vector calculation method was applied to increase effectiveness of the algorithm.

2.1. Multi-objective case

In their work [9], Kachitvichyanukul and Sitthitham proposed two-stage GA, which minimized objective function derived from weighted values of the following criteria: a) maximum processing time of all tasks – makespan, b) total weighted earliness and c) total weighted tardiness. The algorithm is composed of two stages: a) parallel GA searching for the best solution of each individual objective function with migration among populations and b) combining those populations. Authors compared their algorithm with other implementations for each of the used criteria, and for multi-objective case with Multi-Stage parallel GA proposed in [10].

Udomsakdigool and Khachitvichyanukul proposed [22] an ACO algorithm, solving multi-objective JSP (MOJSP) with the sum of weighted normalized values of makespan, mean flow time and mean tardiness as an objective function. Ants use different heuristic information based on priority dispatching rule to diversify the search. Also, local search is applied to intensify the search. Tests were conducted on instances of small sizes, and results were compared with the optimal solutions. Proposed algorithm provided competitive results.

Ripon *et al.* in [16] studied different solution representations for JSP and proposed new crossover operator, called improved precedence preservation crossover (IPPX). Different crossover methods were implemented for non-dominated sorting GA (NSGA-II) [2]. Objective function was composed from mean flow-time and makespan criteria. Proposed approach proved to be able to find the near-optimal solutions with better results and reduce the execution time significantly.

Lei [11] presented a Particle Swarm Optimization (PSO) for the MOJSP in. Objective was to minimize makespan and total job tardiness criteria simultaneously. The global best position selection is combined with crowding-measure-based archive maintenance to design a Pareto archive. Proposed algorithm was capable of producing a high-quality Pareto front.

Multi-objective PSO (MOPSO) was proposed by Sha and Lin [18]. The paper considered following objectives: a) makespan, b) total tardiness and c) total idle time. Giffler and Thompson [6] heuristic was employed to decode schedule into active solution. Test performed on benchmark instances of small sizes showed, that proposed MOPSO provides superior results.

Vázquez-Rodríguez and Petrovic [23] proposed hybrid Dispatching Rule-based GA (DRGA) and compared their new representation with others. Obtained solution sets were evaluated using hyper-volume indicator from paper [25]. Algorithms using their model based hybrid representation (MHR) obtained higher quality solutions and are more robust to the representation size.

Ripon [15] proposed the Jumping Genes GA (JGGA), a hybrid approach capable of searching for near-optimal and non-dominated solutions with better convergence by simultaneously optimizing criteria of makespan and total tardiness. Compared to other existing heuristic evolutionary scheduling approaches, the proposed hybrid approach obtained superior results.

Suresh and Mohanasndaram [20] developed Pareto archived SA (PASA) method for the MOJSP minimizing makespan and mean flow time criteria. PASA made use of both Pareto dominance and a simple aggregating function to accept the candidate solution among the neighbourhood set of solutions generated by the segment random insertion (SRI) neighbourhood structure. Performance of PASA is better compared to other algorithms considered in Suresh and Mohanasndaram's paper.

Lei and Wu designed [12] a crowding-measure-based multi-objective evolutionary algorithm (CMOEA), which made use of the crowding-measure to adjust the external population and assign different fitness for individuals. Proposed algorithm performed well in JSP with two objectives (makespan and total tardiness) minimization.

Other SA approach was proposed by Fattahi *et al.* [4]. They tackled MOJSP with makespan and total weighted tardiness criteria. A scalar approach was used to convert the multi-objective problem to a single objective problem. Unfortunately, results were not compared with other (meta)heuristics.

Flexible JSP was considered by Xiong *et al.* in [24]. Authors study robust scheduling for a flexible JSP with random machine breakdowns. Xiong *et al.* addressed this problem using multi-objective EA.

Our previous works include solving multi-criteria problems using following algorithms: SA for flow shop scheduling [14], modified NSGA-II [2] for network scheduling and load balancing [17] and an implementation of parallel TS for vehicle routing problem [8]. In this paper we combined multi-criteria decision analysis (MCDA) with population-based algorithms in order to achieve an improved methods for solving multi-objective scheduling problems. This approach to MOJSP has not been widely studied in the literature so far.

3. Problem Description

We consider a manufacturing system consisting of m machines given by the set $M = \{1, \ldots, m\}$. The system is to process n jobs given by the set $J = \{1, 2, \ldots, n\}$. The j-th job requires the sequence of n_j operations indexed consecutively $(l_{j-1} + 1, ..., l_{j-1} + n_j)$, where $l_j = \sum_{i=1}^{j} n_i$, is the total number of operations of the first j jobs, $j = 1, 2, ..., n$, $(l_0 = 0)$. The set of all operations is denoted by O and the set of all operations for job j is denoted $\mathcal{O}_j$ $(\mathcal{O}_j \subset O)$.

Operation $x \in O$ is to be processed on machine $\mu_x \in M$ during an uninterrupted processing time $p_x > 0$, $x \in O$. Our aim is to find the schedule under the following constraints: (1) each machine can process at most one product at a time, (2) each product can be processed by at most one machine at a time, (3) operations cannot be preempted. The benchmarks existing in the literature usually assume that $n_j = m$ (*i.e.* each job consists of exactly m operations) and that each operation in a given job requires different machine.

The set of operations O can be decomposed into subsets $O_k = \{x \in O | \mu_x = k\}$, where O_k contains the operations to be processed on machine $k \in M$. Let permutation π_k define the processing order of operations from the set O_k on machine k, and let Π_k be the set of all permutations on O_k. Let Π denote the product of all permutations sets of all machines *i.e.*

$\Pi = \Pi_1 \times \Pi_2 \times ... \times \Pi_m$. The processing order of all operations on machines is determined by m-tuple $\pi = (\pi_1, \pi_2, ..., \pi_m)$, where $\pi \in \Pi$. Our aim is to find a schedule π^* that is both feasible and optimal with regard to the given goal function $f(\pi)$. *i.e.*:

$$\pi^* = \min_{\pi} f(\pi), \qquad \pi^*, \pi \in \Pi_{\text{feas}}, \qquad (1)$$

where $\Pi_{\text{feas}} \subset \Pi$ is the set of feasible schedules.

In order to determine whether a given schedule is feasible we define predecessors and successor for every operation. For any operation $j \in O$ and processing order given by π we define the machine predecessor (successor) $\underline{s}_j$ ($\overline{s}_j$) and technological predecessor (successor) $\underline{t}_j$ ($\overline{t}_j$).

We can describe a schedule of processing jobs for fixed processing order π by vectors $S = (S_1, \ldots, S_o)$ and $C = (C_1, \ldots, C_o)$, where values S_j and C_j denote starting time of operation j and its completion time. The schedule has to satisfy the following constraints:

$$C_{\underline{t}_j} \leq S_j \qquad \underline{t}_j \neq 0, \;\; j \in O, \qquad (2)$$

$$C_{\underline{s}_j} \leq S_j \qquad \underline{s}_j \neq 0, \;\; j \in O, \qquad (3)$$

$$C_j = S_j + p_j \qquad j \in O, \qquad (4)$$

The schedule is feasible if there exists a solution to the inequalities at(2–4). Constraint (2) follows from technological processing order of operations inside job, whereas (3) from the unit capacity of machines. Equation (4) is obvious.

Table 1. Sample instance of JSP

job	operation	machine	execution time
	1	2	2
1	2	3	1
	3	1	3
	4	3	3
2	5	1	2
	6	2	1
	7	1	3
3	8	2	2
	9	3	5
	10	3	1
4	11	1	3
	12	2	2

For example, let us consider JSP instance shown in Tab 1. Here we have 4 jobs to schedule on 3 machines and 12 operations in total. An example of a feasible solution is shown in Fig. 1 with numbers in the figure corresponding with the numbers of operations. Take job 3 for example, which is composed of operations 7, 8 and 9. Operation 7 is placed on machine 1 right away. Operation 8 cannot start its processing until both its machine and technological predecessors (operations 1 and 7 respectively) have completed their execution. In result it waits for operation 7 to complete. Similarly, operation 9 has to wait for its machine (4) and technological (8) predecessors, so it cannot start execution sooner than operation 4 have been completed.

For our research with the multi-criteria variant of JSP problem we chose to consider two optimization criteria: the sum of completion times for all jobs C_{sum} and the maximum completion time of all jobs $C_{\max}$:

$$C_{\text{sum}} = \sum_{j=1}^{n} C_{l_{j-1}+n_j}, \qquad (5)$$

$$C_{\max} = \max_{j \in J} C_{l_{j-1}+n_j}, \qquad (6)$$

where $C_{l_{j-1}+n_j}$ is the completion time of the last operation of job j. Using the example instance from below $C_{\text{sum}} = C_3 + C_6 + C_9 + C_{12} = 42$ and $C_{\max} = \max\{C_3, C_6, C_9, C_{12}\} = 14$.

Because of the multi-criteria nature of the problem our aim is to obtain the set of all Pareto-optimal solutions (instead of a single solution as in Eq. 1), according to the concept of Pareto-efficiency described later on.

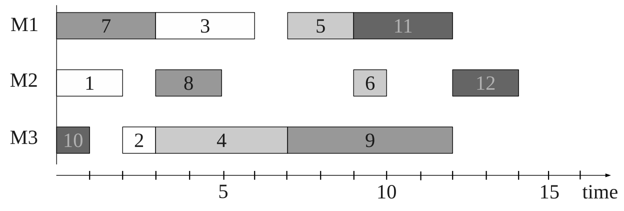

Figure 1. Example of a feasible schedule for the instance from Tab. 1.

4. Evaluation of Multi-Criteria Solutions

Evaluation of multi-criteria solutions is not as straight-forward as single criterion evaluation and comparing two solutions requires different approach. Aggregation of (weighted) objectives is one of the most commonly used techniques, unfortunately this method requires either fine tuning of the weights or running the algorithm with multi-start. Thus, we employed technique from MCDA to evaluate solutions in proposed multi-agent algorithms.

4.1. Technique for order of preference by similarity to ideal solution

Hwang and Yoon proposed TOPSIS, a MCDA method in [7]. The concept for this method is choosing solution that should the shortest geometric distance from the best (ideal) solution and the longest distance from the worst (negative-ideal) solution.

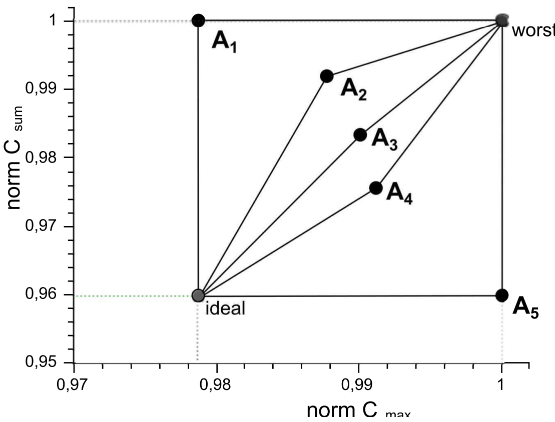

Figure 2. Visualization of the TOPSIS method.

See Fig. 2 for a sample of such distances and selection of worst and best values when minimizing two objective functions. This method uses weights for each criterion and normalizes all solutions before calculating the geometric distance between each of them. The higher the value of relative closeness the better the solution. This method allows to choose one solution from the Pareto front, without involving decision-maker in the process.

4.2. Pareto efficiency

The solution to a multi-objective problem is the set of non-dominated solutions called the Pareto front, 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 $\exists_i \in \{1...n\}$, $y_i < z_i$.

5. Representation and Decoding

In every algorithm development, first thing to do is to decide on solution representation. When solving JSP, it is also important to decide on method for decoding said representation. We employ a job representation in the form of a permutation of jobs, therefore it is a n-element permutation. In this form the obtained solutions are always represented by n numbers regardless of the number of machines m or the number of operations in job j (n_j).

In order to calculate the values of our objective functions, we need a deterministic transformation from a representation into its corresponding solution. Let us assume that we are using the instance from Tab. 1 and that our representation is $R = (2, 1, 4, 3)$. Our decoding procedure works as follows. We create a sequence S_1 by taking the first operations of all jobs (in the example from Tab. 1 those would be operations 1, 4, 7 and 10). The order of operations in S_1 must conform with the order in the representation R, so $S_1 = (4, 1, 10, 7)$. Let us create empty sequence S and add S_1 at its end. Next we create sequence S_2 from the second operations of each job (2, 5, 8 and 11) in order determined by our representation, so $S_2 = (5, 2, 11, 8)$ and we add S_2 at the end of S. We continue this until all operations from O are in S. In result:

$$S = S_1|S_2|S_3 = 4, 1, 10, 7, 5, 2, 11, 8, 6, 3, 12, 9.$$

Now we construct our schedule by inserting elements of S into the best possible place at the moment of insertion (greedy algorithm). Therefore, operation 4 is placed first (in empty schedule) and operation 9 is placed last. Each operation can be placed either at one of the gaps in its target machine (provided that such gaps exist and are large enough for the operation to fit) or after the current last operation of that machine. Moreover, each operation must be placed as to start executing only after its technological predecessor has completed (placement of operation 9 depends on the placement of operation 8, operation 10 can be placed in first suitable gap, as it has no technological predecessors).

The proposed decoding scheme always creates a left-compact schedule, *i.e.* a schedule where no operation can have its starting time decreased without making the schedule infeasible. Moreover, all schedules created using this decoding method are feasible. However, there is no guarantee that the obtained schedule is active and therefore the optimal solution might be unobtainable by this decoding.

For the above method the number of different representations for JSP with n jobs is equal to $n!$ (the number of possible permutations of n elements). Let us consider, for example, the Fisher-Thompson instance ft10 (10 jobs, 10 machines, 100 operations), which has $(10!)^{10} \approx 10^{65}$ possible schedules using the operational representation. Moreover, only approximately 1 in 10^{17} of those schedules is feasible, meaning roughly 10^{48}

feasible solutions [13]. For comparison the number of solutions with our representation equals $n! = 10! = 3628800$, so well below 10^7. This greatly reduces the search space (by 10^{41} in the case of ft10), though only a fraction of the original space is obtainable.

Finally, the chosen representation method should have low redundancy, *i.e.* the number of representations mapping to the same schedule should be as low as possible. The best possible situation is zero redundancy, where each schedule is mapped only by one representation. In order to measure the reduncancy of our representation, we performed a simple computer simulation. 1 000 000 different representation were generated for the TA01 instance (15 jobs, 15 machines). Each representation was decoded and the resulting schedule was stored in a hash map with representations serving as keys and schedules as values. Thus, the number of the unique schedules is equal to the number of elements of the hash map. This test was performed many time over at random and each time the size of the hash map was equal to 1 000 000, therefore we obtained million different schedules from million of different representations. This strongly suggests that our representation and decoding scheme exhibits redundancy equal to or near 0.

6. Proposed Methods

For the purpose of this article two approaches, namely ACO and GA, were proposed and implemented. Both are multi-agent metaheuristics inspired by nature and allow fast Pareto front approximation. Below we present their background, as well as modifications done in order to adapt those algorithms to the MOJSP.

6.1. Ant colony optimization

ACO is a probabilistic metaheuristic technique used to create approximate algorithms for optimization problems. The technique itself was first proposed by Dorigo [3] in order to find good (short) paths in a given graph. Is is currently used for a wide range of discrete optimization problems and can be applied to any problem that can be reduced to short path search in a graph.

ACO is a multi-agent metaheuristic as it is based on the foraging behaviour of ant colonies. Every ant follows a simple procedure: by wandering in half-random directions, it searches for food sources and then lays a chemical agent called pheromone that attracts other ants. The strength of the pheromone indicates the quality of the path marked by it. In result the pheromone serves as a means of communication between ants. The pheromone affects the probability of choosing a given path, therefore attracting ants to converge on the most promising paths. The existing pheromone evaporates with time, allowing the ants to search different possible paths. Even though each ant is a simple agent, their colony shows signs of swarm intelligence, which serves to produce the final solution. Additional advantages of ACO are high potential for parallelization and adaptive abilities, finding new paths when current ones become unusable, due to the pheromone evaporation mechanism.

The implementation of this technique models pheromone in the form of a pheromone matrix. Following an example of a travelling salesman problem (TSP), each ant in a given iteration of the algorithm constructs a feasible solution (a path in the graph). This is done by choosing the next vertex and adding it to the current partial solution. The decision is made with probability based on the pheromone τ lying on a given edge and the visibility η of a given vertex. The visibility of each vertex depends on the problem. In case of the TSP, it is usually a reciprocal of distance (edge weight) between vertices: $\eta = \frac{1}{d}$. The weight of each factor (pheromone and visibility) is adjusted with parameters α and β. The final probability of choosing to go to vertex j from vertex i is given as:

$$p_{i,j} = \begin{cases} \frac{\tau_{i,j}^{\alpha} \cdot \eta_{i,j}^{\beta}}{P_i} & \text{if partial solution is feasible,} \\ 0 & \text{otherwise,} \end{cases} \tag{7}$$

where $P_i = \sum_{n \in N_i} \tau_{i,n}^{\alpha} \cdot \eta_{i,n}^{\beta}$ and N_i are vertices that vertex i connects to.

After each iteration the generated solutions are evaluated and the results are used to perform a pheromone laying procedure. The amount of pheromone used, how many ants are allowed to lay pheromone and several other factors are different depending on the specific version of ACO used. The existing pheromone in the matrix is decreased (evaporated) before the next iteration starts. The quantity of pheromone deposited and evaporated is affected by parameters Q and ρ respectively:

$$\tau_{i,j} = \rho \cdot \tau_{i,j} \tag{8}$$

$$\Delta\tau_{i,j}^{\text{best}} = \begin{cases} Q/L^k & \text{if } (i,j) \text{ belongs to a tour,} \\ 0 & \text{otherwise,} \end{cases} \tag{9}$$

where L^k is the length of the tour found by the kth ant. Equation 9 is applied for every ant.

Many different version of ACO exist. The one used in this paper is based on the Max-Min Ant System (or MMAS) proposed by Stützle and Hoos [19], which proved to be one of the most efficient of developed ACO systems. The most important characteristics of MMAS are: 1) the values of pheromone matrix are bounded from below and above by $\tau_{\min}$ and $\tau_{\max}$ respectively, 2) only one ant in an iteration is allowed to lay pheromone, 3) the pheromone matrix is initialized to $\tau_{\max}$ in the beginning. This characteristics result in better search of solution space in early iterations, reduction of the algorithm stagnation effect and improvement of the search process by elitist pheromone laying strategy. In their original work Stü tzle and Hoos assume that $\tau_{\max}$ and $\tau_{\min}$ can change between iterations and are connected by equation:

$$p_{\mathrm{dec}} = \sqrt[n]{p_{\mathrm{best}}} = \frac{\tau_{\max}}{\tau_{\max} + (\mathrm{avg} - 1)\tau_{\min}}. \qquad (10)$$

Therefore $\tau_{\min}$ can be derived given the values of $\tau_{\max}$ and probability p_{dec}. More information on the desired value of p_{dec} can be found in the original paper by Stützle and Hoos.

In order to adapt the original MMAS algorithm for MOJSP, a few modifications were performed. First, we needed a connection between JSP and a graph search problem. Fortunately, the chosen solution representation is, in fact, a permutation of jobs and corresponds well to the permutation of cities, which is a solution to the traveling salesman problem. The only difficulty lies in the visibility $\eta_{i,j}$ for each pair of jobs. We have chosen a heuristic in which the weight of each edge connecting to the vertex (job) j is the sum of processing times of all operations of j:

$$\eta_{i,j} = \eta_j = \sum_{k \in \mathcal{O}_j} p_{k,j}, \qquad (11)$$

where $\mathcal{O}_j$ is set of operations for job j. In result, the visibility depends only on the "target" job and not on the "source" job. This heuristic is based on the job inserting techniques from the classic scheduling algorithms.

The original MMAX is a single-criterion algorithm. In order to deal with multiple-criteria we employ the TOPSIS method described above for evaluation of solutions, coupled with the decoding of constructed solutions. The pheromone update can be performed based on distance obtained by TOPSIS, as well as the C_{sum} and $C_{\max}$ criteria directly. For the use of TOPSIS method the weights for criteria were defined. Since our goal is to obtain the set of Pareto-efficient solutions we add each constructed solution to the Pareto set and remove all dominated solutions after each iteration. Moreover, we use certain constructive algorithm in the first iteration in order to obtain good starting solutions. Since the algorithm is a single criterion method, half of the ants construct the solution based on C_{sum}, while the other half uses $C_{\max}$. On the second and subsequent iterations ants construct their tours as in regular Max-Min Ant System.

6.2. Genetic algorithm

GA is a multi-agent metaheuristic, which uses evolution to find better solutions [2]. Evolutionary algorithms use techniques inspired by natural evolution, such as inheritance, mutation, selection, and crossover to generate solutions to optimization problems. Usually the evolution starts from random initial population, which is a set of specimens. In each iteration, called a generation, the specimens are modified with the use of genetic operators (namely crossover and mutation) and their fitness is evaluated in order to select best solutions for the next generation. Many different approaches to GA were proposed and tested for a variety of optimization problems.

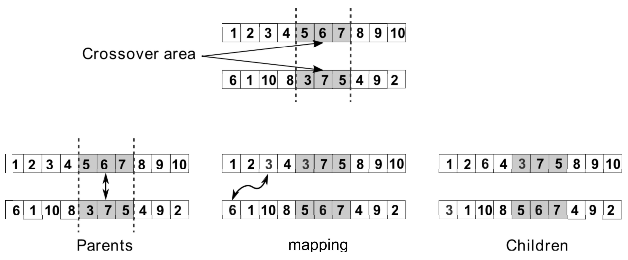

Figure 3. Partially matched crossover example.

GA implemented for the purpose of this papers uses Pareto archive, in order to retain non-dominated solutions through successive iterations. The individuals in population are represented by jobs permutation, values of criteria functions and relative closeness indicator is supplied by the TOPSIS method. Initial population includes solutions obtained from constructive single-criterion algorithms (applied for each criterion). Such initialization allows faster designate approximation of the Pareto front. Mutation is performed by multiple interchanging two random jobs in schedule, while crossover used is partially matched crossover (PMX) (see Fig. 3) scheme. It randomly cuts chromosomes (permutations of jobs) at two points and interchanges middle parts between individuals. In order to retain feasibility of solutions, repeated jobs are mapped and permutations are repaired. Fitness of each solution is evaluated using TOPSIS technique. Tournament selection allows us to choose individuals for next iteration of the algorithm from both parent and child populations. Comparing two populations of the same size as output population leads to discarding half of previous solutions. Moreover, when relative closeness values converge to zero, an anti-stagnation is employed. It takes non-dominated solutions from Pareto archive and copies them (multiple times if their number doesn't exceed the number of individuals in population) to parent population before employing mutation and crossover operations on said population. Such action allows the algorithm to restart genetic search from good solutions.

7. Computer Experiment

All algorithms were implemented in C++ programming language and compiled with Visual Studio 2010. The programs were tested on Intel Core i7-3770 3400MHz, 8GB RAM and running Windows 7 64b. We use a set of benchmarks from literature [21] that consist of 8 groups (combined by instance size), 10 instances per group. The ACO method was run using following parameters: $\alpha = 1.0$, $\beta = 2.0$, $\rho = 0.8$, $p_{\text{best}} = 0.8$, number of ants $a = 200$ and number of iterations $i = 10\,000$.

7.1. Multi-criteria quality indicators

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 and the concept of the Pareto front. The solutions in the front do not dominate each other, but the solutions outside of the front are always dominated by at least one solution in the front. However, we use approximate algorithms, meaning that the obtained non-dominated solution sets may be different from the optimal Pareto front and different for each algorithm. We need an indicator for quality-based comparison of the obtained fronts. A number of such methods exists, out of which we have chosen following.

Aggregated Pareto solutions. Certain method of comparison of Pareto front approximations was devised in paper [14]. This approach creates a set of all non-dominated solutions found by all algorithms and calculates the percentage of the non-dominated solutions found by the specific algorithm in question which then serves as an indicator. Solutions from all algorithms were flagged and combined 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.

Hyper-volume indicator. Zitzler *et al.* [25] provided a few necessary tools for a better evaluation and comparison of multi-objective algorithms. They proposed, among others, a hyper-volume indicator I_H to measure quality of the Pareto front approximations. Hyper-volume indicator measures the area covered by the approximated Pareto fronts for each of 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.

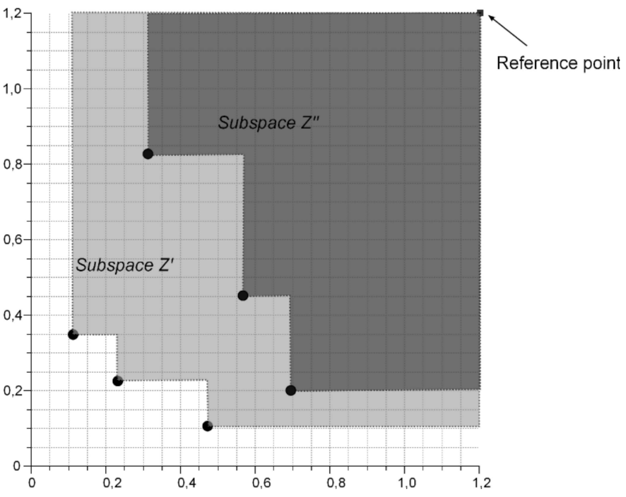

Figure 4. Visualization of hyper-volume indicator.

In our case, reference points were calculated as follows. For each of the criteria used we took the worst value from both non-dominated sets, multiplied it by 1.2 and assigned as reference points value for that criterion. Visualization of such indicator is presented in Fig. 4.

7.2. Results

There are 80 instances divided into 8 instance sizes, thus computation results were combined into groups. Said groups are of following sizes: 15×15, 20×15, 20×20, 30×15, 30×20, 50×15, 50×20 and 100×20. Computation times were similar for both algorithms, in order to maintain close test conditions.

For each test instance and for each run of algorithms, we collected the following values:

- $|P|$ – number of non-dominated solutions in aggregated approximations of Pareto fronts from all algorithms,
- $|P_x|$ – number of non-dominated solutions found using algorithm x,
- $|D_x|$ – number of unique Pareto solutions found using algorithm x,
- Q – quotient of I_H computed for GA to I_H computed for ACO,

where $x \in \{\text{ACO}, \text{GA}\}$. Due to characteristics of evolutionary algorithm and ant colony, numbers of unique Pareto and non- dominated solutions are the same. Solutions found by both algorithms are disjoint.

We evaluated number of non-dominated solutions and hyper-volume indicator for each instance. Summed up numbers of Pareto solutions for each instance size can be seen in Tab. 2. Number of Pareto solutions found by ACO exceeds the number of solutions found by GA, which only found around 55% of the number of non-dominated solutions provided by ACO. Moreover, in Tab. 2 we placed average values of hyper-volume indicator evaluated for each group. ACO proved to provide superior results to GA in both the number of non- dominated (and unique) solutions, as well as the value of I_H.

Table 2. No. of Pareto solutions and Hyper-volume indicator values.

Group	$\|P\|$	$\|P_x\|$		$\|D_x\|$		Q
		GA	ACO	GA	ACO	
15×15	104	37	67	37	67	0,77
20×15	118	39	79	39	79	0,74
20×20	97	35	62	35	62	0,83
30×15	119	38	81	38	81	0,79
30×20	131	50	81	50	81	0,85
50×15	172	71	101	71	101	0,55
50×20	150	49	101	49	101	0,67
100×20	169	65	104	65	104	0,54

In the process of results analysis we found out that none of the solutions provided by one of algorithms were dominated by the other. Moreover, all solutions were unique, meaning each solution was found only by one of the proposed algorithms. Although both algorithms employed the same evaluation method, namely TOPSIS, and used the same criteria weights, the evolutionary approach tended to minimize makespan while swarm intelligence provided better values of total flow time objective. This feature might be a consequence of using pheromone update matrix and visibility used in ACO algorithm, which led the agents (ants) to follow trails that provide lower values of total flowtime objective, while individuals in GA inherit properties from their parents and, during evolution, attempt to improve values of both objectives. Fig. 5 presents an example of such behaviour. Moreover, difference in the number of non-dominated and unique Pareto solutions might be caused by properties of tested algorithms.

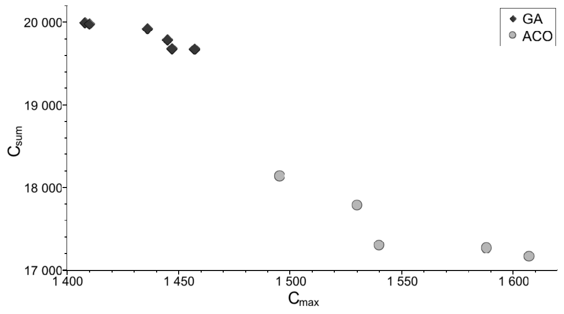

Figure 5. Pareto front aggregated from GA and ACO algorithms for TA04.

8. Conclusions and Further Research

JSP is considered a complex and important discrete combinatorial optimization problem. Multi-criteria scheduling, and especially JSP, is still growing field of optimization problems. Relatively small number of papers concerning MOJSP and only a few of algorithms were proposed. This paper proposes new look at scheduling model widely used in real production problems. We proposed and tested two nature-based algorithms, which provided some unusual results. Even though the same representation and decoding scheme were used, evolutionary methods and swarm intelligence explore other areas of the approximation of Pareto front in solutions space. We concluded, that those differences are result from other methods of storing information about solutions. It impacts the results our algorithms provided in a way, that collections of non-dominated solutions supplement each other.

Our further research will use more complex solution representation, i.e. operation based representation, and new encoding/decoding schemes. Different representation will allow us to explore bigger or different solution space, while other decoding schemes can be more effective for other objective functions.

9. Acknowledgements

The work was co-financed by the *Młoda Kadra*, grant no. B40112.

References

[1] Bożejko W., Pempera J., Smutnicki C., Parallel Simulated Annealing for the Job Shop Scheduling Problem Lecture Notes in Computer Science, Vol. 5544, pp 631–640 (2009).

[2] Deb K., Pratap A., Agarwal S., Meyarivan T., A fast and elitist multi–objective genetic algorithm: NSGA–II, IEEE Trans. EVol. Comput., Vol. 6 (2), pp 182–197 (2002).

[3] Dorigo M., Maniezzo V., Colorni A., Ant System: Optimization by a colony of cooperating agents, IEEE Transactions on Systems, Man, and CyberneticsPart B, Vol. 26 (1), pp 29–41 (1996).

[4] Fattahi P., Saidi Mehrabad M., Arynezhad M. B., An algorithm for multi objective job shop scheduling problem, Journal of Industrial International, Vol. 2 (3), pp 43–53 (2006).

[5] Garey M., Johnson, D., Computers and Intractability: A Guide to the Theory of NP–Completeness, W. H. Freeman & Co. New York, USA, (1979).

[6] Giffler B., Thompson G. L., Algorithms for Solving Production–Scheduling Problems, Operations Research, Vol. 8 (4), pp 487–503 (1960).

[7] Hwang C.L., Yoon K., Multiple Attribute Decision Making: Methods and Applications, Springer–Verlag, New York (1981).

[8] Jagiełło S., Żelazny D., Solving multi-criteria vehicle routing problem by parallel tabu search on GPU, Procedia Computer Science, Vol. 18, pp 2529–2532 (2013).

[9] Kachitvichyanukul V., Sitthitham S., A two–stage genetic algorithm for multi–objective job shop scheduling problems, Journal of Intelligent Manufacturing, Vol. 22 (3), pp 355–365 (2011).

[10] Lam N. V., Kachitvichyanukul V., Luong H. T., A multi–stage parallel genetic algorithm for multi–objective job shop scheduling, The 6th Asia Pacific Industrial Engineering and Management Systems Conference (APIEMS 2005), Philippines (2005).

[11] Lei D., A Pareto archive particle swarm optimization for multi–objective job shop scheduling, Computers and Industrial Engineering, Vol. 54 (4), pp 960971 (2008).

[12] Lei D., Wu Z., Crowding–measure–based multi–objective evolutionary algorithm for job shop scheduling, International Journal of Advanced Manufacturing Technology, Vol. 30, pp 112–117 (2006).

[13] Nowicki E., Smutnicki C., Some new ideas in TS for job shop scheduling, Metaheuristic optimization via memory and evolution. Tabu search and scatter search. Kluwer Academic Publ., pp 165-190 (2005).

[14] Pempera J., Smutnicki C., Żelazny D., Optimizing bicriteria flow shop scheduling problem by simulated annealing algorithm, Procedia Computer Science, Vol. 18, pp 936-945 (2013).

[15] Ripon K. S. N., Hybrid evolutionary approach for multi–objective job shop scheduling problem, Malaysian Journal of Computer Science, Vol. 20 (2), pp 183–198 (2007).

[16] Ripon K. S. N., Siddique N. H., Torresen J., Improved precedence preservation crossover for multi–objective job shop scheduling problem, Evolving Systems, Vol. 2 (2), pp 119–129 (2011).

[17] Rudy J., Żelazny D., Memetic algorithm approach for multi-criteria network scheduling, Proceeding of the International Conference on ICT Management for Global Competitiveness and Economic Growth in Emerging Economies, pp 247–261 (2012).

[18] Sha D. Y., Lin H–H., A multi–objective PSO for job–shop scheduling problems, Expert Systems with Applications, Vol. 37 (2), pp 1065–1070 (2010).

[19] Stützle T., Hoos, H. H., MAXMIN Ant System, Future Generation Computer Systems, Vol. 16 (8), pp. 889–914 (2000).

[20] Suresh R.K., Mohanasndaram K.M., Pareto archived simulated annealing for job shop scheduling with multiple objectives, International Journal of Advanced Manufacturing Technology, Vol. 29, pp 184 –196 (2006).

[21] Taillard E., Benchmarks for basic scheduling problems, European Journal of Operational Research, Vol. 64, pp 278-285 (1993).

[22] Udomsakdigoola A., Khachitvichyanukul V., Ant colony algorithm for multi–criteria job shop scheduling to minimize makespan, mean flow time and mean tardiness, International Journal of Management Science and Engineering Management, Volume 6 (2), pp 117–123 (2011).

[23] Vázquez–Rodríguez J. A., Petrovic S., A new dispatching rule based genetic algorithm for the multi–objective job shop problem, Journal of Heuristics, Vol. 16 (6), pp 771–793 (2010).

[24] Xionga J., Xinga L-N., Chena Y-W, Robust scheduling for multi-objective flexible job-shop problems with random machine breakdowns, International Journal of Production Economics, Vol. 141(1), pp 112–126 (2013).

[25] Zitzler E., Thiele L., Laumanns M., Fonseca C. M., Fonseca V., Performance assessment of multi–objective optimizers: An analysis and review, IEEE Trans. Comput., Vol. 7 (2), pp 117–132 (2003).

Jarosław Rudy *Author is a Ph.D. Student at Wrocław University of Technology where he received his M.Sc. in computer science and an works as an assistant at the Institute of Computer Engineering, Control and Robotics at the same university. His teaching classes focus on operating systems and programming languages. His research interests include discrete and multi-criteria optimization (scheduling in particular), operations research, computability theory and models of computation. He is the author and co-author of several papers, some of which appeared in Journal of Applied and Theoretical Computer Science, Journal of Applied Computer Science and in the Web of Science.*

Dominik Żelazny *Author is a Ph.D. student at Wrocław University of Technology. He teaches discrete optimization, information technology and control of production processes. His research interests are scheduling problems, multi-objective optimization, vehicle routing problems, multi-criteria decision making and other discrete optimization problems. He is the co-author of chapter in book monography Discrete optimization in Computer Engineering, Control and Robotics. His paper appear in numerous conferences including International Conference on Computational Science, International Conference on ICT Management for Global Competitiveness & Economic Growth in Emerging Economies, Production engineering : innovations and technologies of the future, Innovations in Management & Production Engineering and National Conference of Discrete Processes Automation. His papers appeared in journal Procedia Computer Science.*