Technical Sciences, 2019, 22(1), 61–74

UNIWERSYTET WARMIŃSKO-MAZURSKI W OLSZTYNIE

Quarterly peer-reviewed scientific journal
ISSN 1505-4675
e-ISSN 2083-4527
TECHNICAL SCIENCES
Homepage: www.uwm.edu.pl/techsci/

PARTICLE SWARM OPTIMIZATION AND DISCRETE ARTIFICIAL BEE COLONY ALGORITHMS FOR SOLVING PRODUCTION SCHEDULING PROBLEMS

Tadeusz Witkowski

Faculty of Production Engineering
Warsaw University of Technology in Warsaw

Received 13 December 2018, accepted 6 February 2019, available online 7 February 2019.

Key words: Discrete Artificial Bee Colony, Particle Swarm Optimization, production scheduling problem, makespan.

Abstract

This paper shows the use of Discrete Artificial Bee Colony (DABC) and Particle Swarm Optimization (PSO) algorithm for solving the job shop scheduling problem (JSSP) with the objective of minimizing makespan. The Job Shop Scheduling Problem is one of the most difficult problems, as it is classified as an NP-complete one. Stochastic search techniques such as swarm and evolutionary algorithms are used to find a good solution. Our objective is to evaluate the efficiency of DABC and PSO swarm algorithms on many tests of JSSP problems. DABC and PSO algorithms have been developed for solving real production scheduling problem too. The experiment results indicate that this problem can be effectively solved by PSO and DABC algorithms.

Introduction

The job shop scheduling problem has become a classic scheduling problem. Although it is mainly associated with industrial engineering, it is required in other sectors, too. The study of the scheduling problem is carried out taking inputs from various sources, such as computer science, operations research, management and manufacturing.

Correspondence: Tadeusz Witkowski, Instytut Organizacji Systemów Produkcyjnych, Wydział Inżynierii Produkcji, Politechnika Warszawska, ul. Narbutta 85, 02-524 Warszawa, e-mail: tawit@poczta.onet.pl

Scheduling involves assigning a set of tasks on resources in a time period, taking into account the time, capability and capacity constraints. Many studies have been done to solve this problem or to determine the closest approach to the solution. Commonly used scheduling techniques include the following (BŁAŻEWICZ et al. 2007):

– Exact Algorithms (e.g. Branch and Bounds Methods, Linear Programming, Dynamic Programming);

– Approximation Algorithms [Artificial Intelligence Algorithms (e.g. Artificial Neural Network), Local Search Algorithms (e.g. Greedy Randomized Adaptive Search Procedure, Taboo Search, Simulated Annealing), Evolutionary Algorithms (e.g. Genetic Algorithm), Swarm Optimization Algorithms (e.g. Ant Colony Optimization, Bee Colony Algorithm, Particle Swarm Optimization)].

WITKOWSKI (2016) presents a research study of state-of-the-art algorithms for FJSP. Therefore, these algorithms can be called swarm-intelligence-based, bio-inspired, physics-based and chemistry-based, depending on the sources of inspiration.

Swarm-intelligence-based algorithms include: Accelerated PSO, Ant colony optimization, Artificial bee colony, Bacterial foraging, Bacterial-GA Foraging, Bat algorithm, Bee colony optimization, Bee system, BeeHive, Wolf search, Bees algorithms, Bees swarm optimization, Bumblebees, Cat swarm, Consultant-guided search, Cuckoo search, Eagle strategy, Fast bacterial swarming algorithm, Firefly algorithm, Fish swarm/school, Good lattice swarm optimization, Glowworm swarm optimization, Hierarchical swarm model, Krill Herd, Monkey search, Particle swarm algorithm, Virtual ant algorithm, Virtual bees, Weightless Swarm Algorithm and other algorithms Anarchic society optimization, Artificial cooperative search, Backtracking optimization search, Differential search algorithm, Grammatical evolution, Imperialist competitive algorithm, League championship algorithm, Social emotional optimization.

Bio-Inspired (not swarm-intelligence-based) algorithms include: Atmosphere clouds model, Biogeography-based optimization, Brain Storm Optimization, Differential evolution, Dolphin echolocation, Japanese tree frogs calling, Eco-inspired evolutionary algorithm, Egyptian Vulture, Fish-school Search, Flower pollination algorithm, Gene expression, Great salmon run, Group search optimizer, Human-Inspired Algorithm, Invasive weed optimization, Marriage in honey bees, OptBees, Paddy Field Algorithm, Roach infestation algorithm, Queen-bee evolution, Shuffled frog leaping algorithm, Termite colony optimization.

Physics- and chemistry-based algorithms include: Big bang-big Crunch, Black hole, Central force optimization, Charged system search, Electro-magnetism optimization, Galaxy-based search algorithm, Gravitational search, Harmony search, Intelligent water drop, River formation dynamics, Self-propelled particles, Simulated annealing, Stochastic diffusion search, Spiral optimization, Water cycle algorithm.

Other algorithms used for job scheduling problem are the following: Raven roosting optimization algorithm, Camel herds algorithm, artificial flora algorithm, Rhinoceros search algorithm, Beer froth artificial bee colony.

Hence, a variety of heuristics and metaheuristics procedures such as taboo search, simulated annealing and genetic algorithm have been applied to solve these problems and find an optimal or near-optimal schedule in a reasonable time (WITKOWSKI et al. 2010) Swarm intelligence systems are typically made up of a population of simple agents interacting locally with one another and with their environment. Particle Swarm, Ant Colony, Bee Colony are examples of swarm intelligence. A survey (KRAUSE et al. 2013) shows that PSO was the most frequently found algorithm, representing 25% of all papers analyzed, and scheduling problems are the most frequently analyzed. The ABC algorithm came second, representing 13% of the total.

Production Scheduling Problem

The structure of the production scheduling problem can be described as follows (MESGHOUNI et al. 2004).

Consider a set of N jobs $\{J_j\}_{i\,j\,N}$, where these jobs are independent of one another; each job J_j has an operating sequence, called G_j; each operating sequence G_j is an ordered series of operations, G_{ij} indicating the position of the operation in the technological sequence of the job; the realization of each operation O_{ij} requires a resource or a machine selected from a set of machines, $\{M_k\}_{i\,j\,M}$ (for FJSSP problem); M is the total number of machines existing in the shop, this implying the existence of an assignment problem; there is a predefined set of processing times; for a given machine, and a given operation O_{ij}, the processing time is denoted by P_{ijMk}; an operation which has started runs to completion (non-preemption condition); each machine can perform operations one after another (resource constraints). Our objective is to determine the minimal makespan ($C_{\max}$ value), where $C_{\max} = \max\{C_j\}$, and C_j is the completion time of job J_j.

To evaluate schedules different performance measures or optimality criteria have been used (BŁAŻEWICZ et al. 2007): schedule length (makespan), mean flow time, mean weighted flow time, maximum lateness, mean tardiness, mean weighted tardiness, mean earliness, mean weighted earliness, number tardy task and weighted number of tardy tasks.

The flexible job shop scheduling problem (FJSSP) is an extended traditional JSSP problem. It discards the restriction of unique resources and allows each operation to be processed by several different machines, and so makes the JSSP problem more similar to the actual production situation.

PSO and ABC algorithms for Solving the Scheduling Problem

The issues of production scheduling cover a wide range of models and algorithms as well as optimization criteria. Papers on JSSP problems with basic PSO and ABC algorithms (non-hybrid) were presented, among others, in SUREKHA and SUMATHI (2010) and ABU-SRHAHN and AL-HASAN (2015). In SUREKHA and SUMATHI (2010), we find a knowledge – based approach to JSSP using Particle Swarm Optimization and Ant Colony Optimization. The well known FISHER and THOMPSON (1963) 10×10 instance (FT10) and ADAMS et al. (1988) 10×10 instance (ABZ10) problem are selected as the experimental benchmark problems. Based on simulation and evaluation results, it is concluded that PSO is the superior computational intelligence algorithm for solving the JSSP problem.

A. ABU-SRHAHN and M. AL-HASAN (2015) present a hybrid algorithm (Cuckoo Search Optimizer is used along with a GA) to minimize $C_{\max}$ value for JSSP. The algorithms were tested using well known datasets in order to verify the validity of the proposed algorithm. The instances FT06, FT10, and FT20 are designed by FISHER and THOMPSON (1963), and instances LA01 to LA16 are designed by LAWRENCE (1984). The results have been compared with GA algorithm and Ant Colony Optimization Algorithm (ACO) to show the importance of the proposed algorithm. The results show that the hybrid algorithm yields the best solutions as measured by $C_{\max}$ value, and GA algorithm is better then ACO.

In this paper, Discrete Artificial Bee Colony (DABC) algorithm and Particle Swarm Optimization (PSO) algorithm are proposed for solving the job shop scheduling problem with the objective of minimizing makespan (which is the total length of the schedule, that is, when all the jobs have finished processing – $C_{\max}$ value).

PSO Algorithm

Particle swarm optimization (PSO) is a population-based optimization algorithm. Each particle is an individual and the swarm is composed of particles. The problem solution space is formulated as a search space. Each position in the search space is a correlated solution of the problem. Particles cooperate to find out the best position (best solution) in the search space (solution space). Particles move toward the pbest position and gbest position with each iteration. The pbest position is the best position found by each particle so far. Each particle has its own pbest position. The gbest position is the best position found by the swarm so far. The particle moves according to its velocity. The velocities are randomly generated toward pbest and gbest positions. For each particle k and dimension j, the velocity and position of particles can be updated by the following equations:

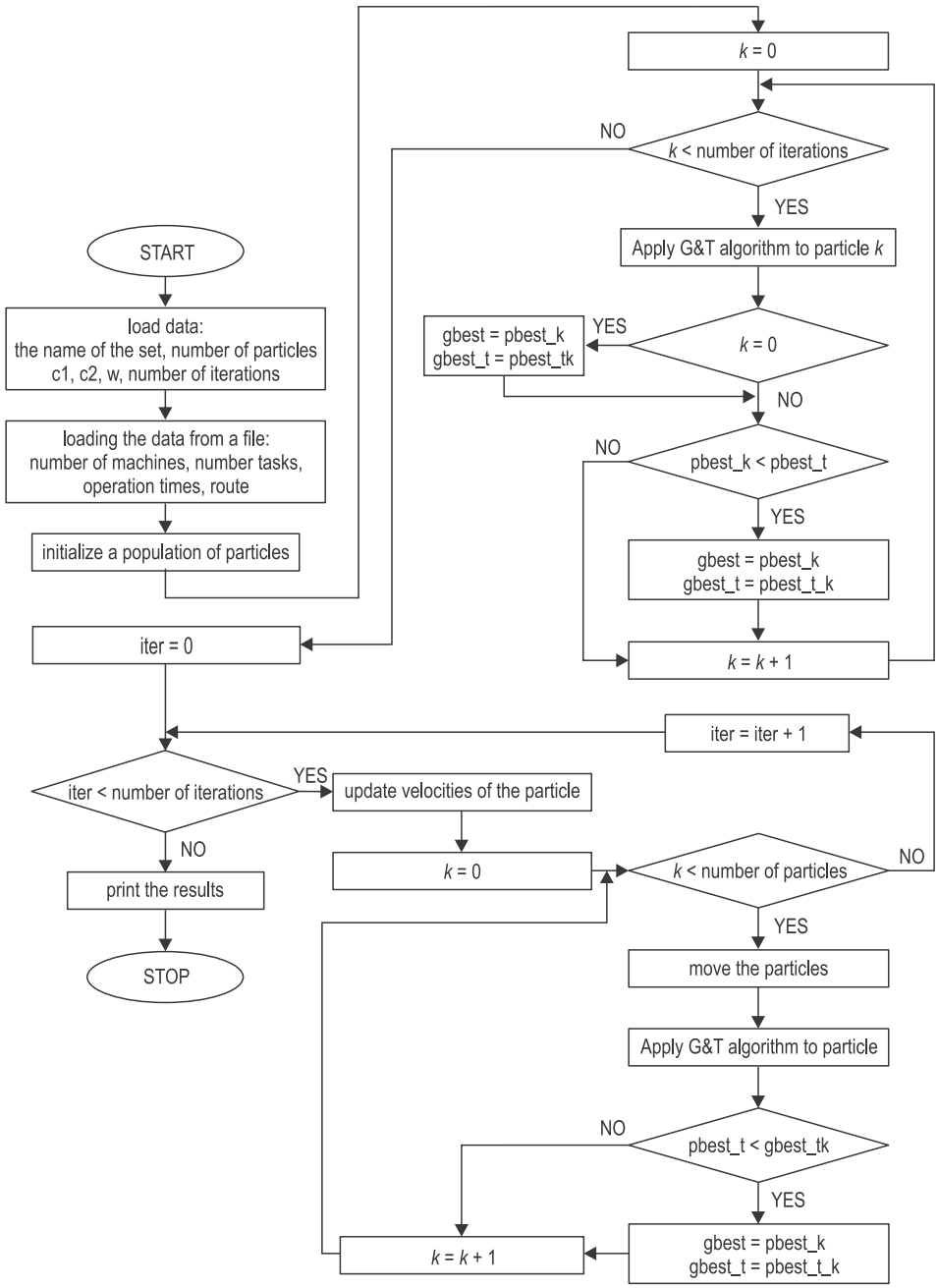

Fig. 1. The detailed flow chart of PSO algorithm to solve JSSP
Source: based on SHUA and HSU (2006).

$$V_{kj} \leftarrow w\, v_{kj} + c_1\, \mathrm{rand}_1\, (\mathrm{pbest}_{kj} - x_{kj}) + c_2\, \mathrm{rand}_2\, (\mathrm{gbest}_j - x_{kj}) \quad (1)$$

$$x_{kj} \leftarrow x_{kj} + v_{kj} \quad (2)$$

In Equations (1) and (2), v_{kj} is the velocity of particle k on dimension j, and x_{kj} is the position of particle k on dimension j. The pbest_{kj} is the pbest position of particle k on dimension j, and gbest_j is the gbest position of the swarm on dimension j. The inertia weight w is used to control exploration and exploitation. The particles maintain high velocities with a larger w, and low velocities with a smaller w. A larger w can prevent particles from becoming trapped in local optima, and a smaller w encourages particles exploiting the same search space area. The constants c_1 and c_2 are used to decide whether particles prefer moving toward a pbest position or gbest position. The rand_1 and rand_2 are random variables between 0 and 1. The process for PSO is as follows (SHUA, HSU 2006):

Step 1: Initialize a population of particles with random positions and velocities on d dimensions in the search space.

Step 2: Update the velocity of each particle, according to Equation (1).

Step 3: Update the position of each particle, according to Equation (2).

Step 4: Map the position of each particle into solution space and evaluate its fitness value according to the desired optimization fitness function. At the same time, update pbest and gbest position if necessary.

Step 5: Loop to step 2 until a criterion is met, usually a sufficiently good fitness or a maximum number of iterations.

Figure 1 shows the detailed flow chart of PSO algorithm to solve JSSP.

DABC Algorithm

The classical artificial bee colony (ABC) algorithm proposed by D. Karaboga (KARABOGA, BASTURK 2007) is a population algorithm often used for constrained optimization problems. As the basic ABC algorithm was originally designed for continuous function optimization, in order to make it applicable for solving the problem considered, a discrete version of the ABC algorithm (DABC). Composite mutation strategies are proposed to enable the DABC to explore the new search space and solve the permutation flow shop scheduling problem. We consider each discrete job permutation as a food source (FS) and apply discrete operations to generate a new neighborhood food source for different bees to make artificial bee colony algorithm suitable for JSP. Each FS is a permutation of operations. In order to generate good diversity neighboring solutions several mutation strategies are proposed to enable the DABC to solve the JSP. Initial population consists of schedule generated based on a random permutation of operations (priority rule). Whenever during employed or onlooker bee phase a new food source (schedule) is produced, for each bee new neighboring solutions are generated using proposed mutations strategies. Figure 2 shows the flow chart of DABC algorithm to solve JSSP.

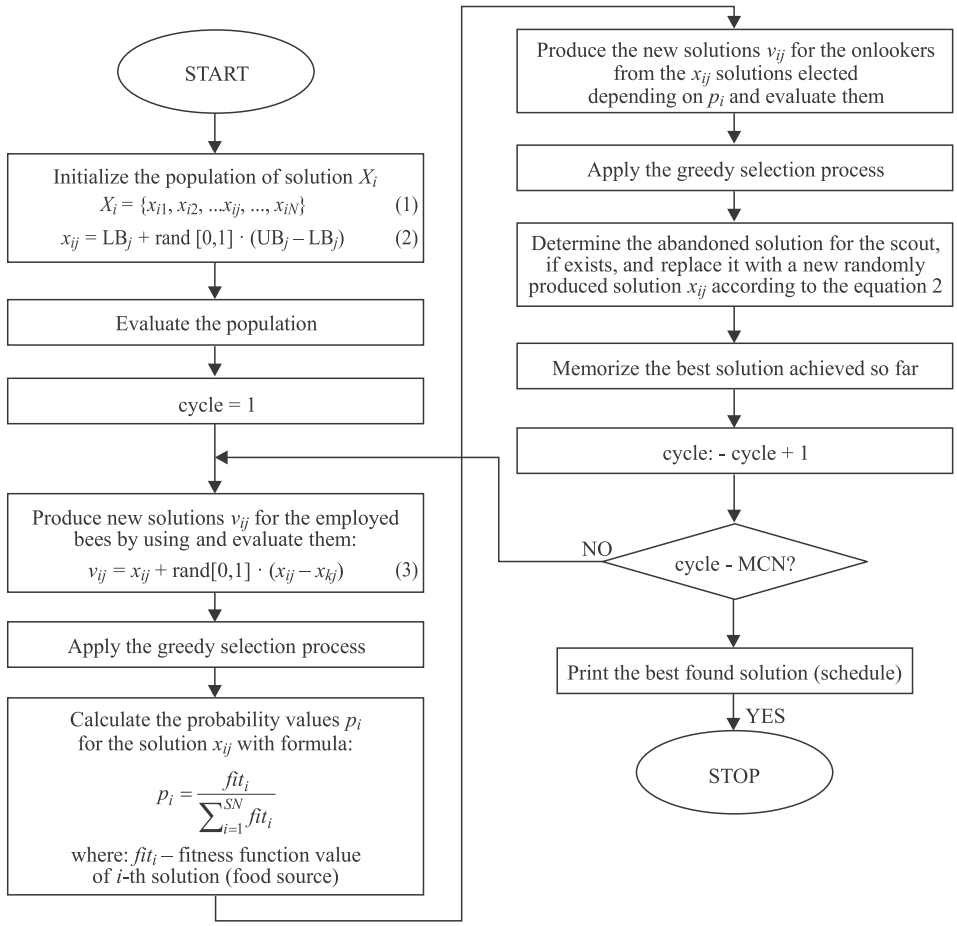

Fig. 2. The flow chart of DABC algorithm to solve JSSP
Source: based on KARABOGA and BASTRUK (2007).

Experiment Results

This section describes the computational experiments to evaluate the performance of the proposed algorithms. The experiments were performed on a PC with a processor Intel® Core™ i7-3770 CPU @ 3.40 GHz and RAM: 16 GB. The DABC algorithm was coded in Java and PSO in C++. We use 10 instances from three classes of standard JSSP test problems: instances FT06, FT10, FT20 (FISHER, THOMPSON 1963), instances La02, La19, La21, La27, La30, La40 (LAWRANCE 1984), and SWV11 instance (STORER et al. 1992).

A computational experiment with PSO for JSSP problems

The JSSP test problems are solved by PSO with the number of particles equaling 30, c_1=0.5, c_2=0.3, w=0.5, and 10, 100, 1,000 iterations (SHUA, HSU 2006). The proposed algorithm is tested on 10 job shop scheduling benchmark problems and the outcomes are presented in Table 1.

Table 1

Performance of PSO algorithm (C_{max} value)

Problem size	Performance of PSO (C_{max} value)			
	optimal makespan BKS	Avg (Best) C_{max} 10 iterations	Avg (Best) C_{max} 100 iterations	Avg (Best) C_{max} 1000 iterations
FT6	55	58.7 (57)	58.2 (57)	57.7 (57)
FT10	930	1,129 (1,075)	1,100 (1,075)	1,070 (1,055)
FT20	1,165	1,313 (1,281)	1,258 (1,215)	1,219 (1,170)
La02	655	743 (704)	707 (685)	668 (664)
La19	842	1,000 (973)	971 (948)	951 (922)
La21	1,046	1,334 (1,303)	1,294 (1,258)	1,266 (1,218)
La27	1,235	1,604 (1,523)	1,570 (1,539)	1,531 (1,490)
La30	1,355	1,648 (1,546)	1,637 (1,605)	1,586 (1,567)
La40	1,222	1,559 (1,538)	1,493 (1,469)	1,462 (1,401)
SWV11	2,983	3,763 (3,668)	3,690 (3,632)	3,651 (3,589)

As can be seen from Table 1, the best C_{max} values were obtained with more iterations. It is obvious that with the increase in the number of iterations we get better results of the criterion function, which, however, is associated with a greater amount of calculations. For example, for the SWV 11 problem, the average calculation time for 10 iterations is 2.01 seconds, while for 1000 iterations it increases to 176.15 seconds. Table 1 shows that the best solutions have been found for FT6, FT20 and La02 tasks. Thus, with the growth in the size of the problem, the efficiency of the algorithm worsens.

A computational experiment with DABC for JSSP problems

Due to several parameters and levels of DABC, full factorial experimental design requires high computational resources and is time consuming because of the large number of experimental runs for each replication. The DABC factors and its combinations (design points) in this work are summarized in Table 2. Those factors are the combination of the Source Number (SN), the Maximum Cycle Number (MCN), and Maximum Improvement Trial Number (MITN).

Table 2

Combinations of SN, MCN, and MINT values used with DABC for solve JSSP

Factor	Combinations (design points)				
	1	2	3	4	5
SN	50	200	500	1000	2000
MCN	1,000	4,000	10,000	20,000	40,000
MITN	200	400	600	800	1,000

We want to determine if the factors interact with one another, i.e., whether the effect of one factor in the response depends on the levels of the others. A number of program start-ups were made for various combinations of factor values, each time changing only one of them. For each combination the program was started five times. Results are presented in Figures 3*a* to 3*c*.

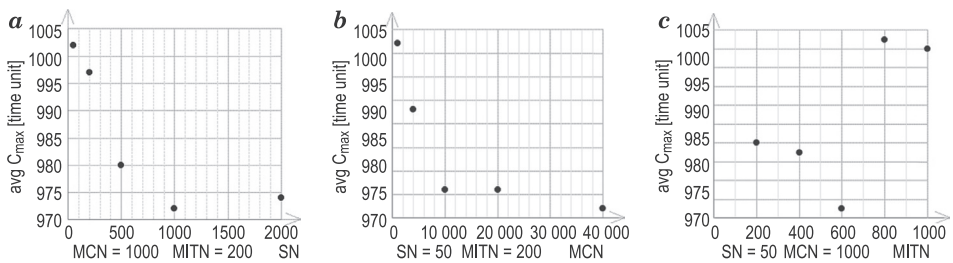

Fig. 3. Influence: *a* – source number (SN), *b* – the maximum cycle number (MCN), and *c* – maximum improvement trial number (MINT) on C_{max} value

Figure 3*a* shows that an increase in the population number (number of food sources – SN) results in achieving a better C_{max} value. With SN =1000, the value reached the lowest average C_{max} value, and no progressive decrease was observed for the average of C_{max} with SN = 2,000 value. Figure 3*b* shows that increasing the maximum cycle number (MCN) results in obtaining a better C_{max} value, and the spread of results decreases with an increase in the population number. We can see an improvement of C_{max} value compared to the previous test point, which is a result of an increase in the MCN value. With MCN = 10,000, the value improvement effect is lower than for previous test points. Figure 3*c* shows that the best C_{max} value was achieved with MITN equaling 60% of MCN value, when the average C_{max} was the lowest.

Figure 4*a* to 4*b* show an influence of other combinations (SN, MCN, and MINT values) on the average and best C_{max} value.

When analyzing Figure 4*a*, we can see that increasing the maximum cycle number (MCN) gives better results (lower C_{max} value) than increasing the source number (SN). For example, for the third test point (optimal C_{max} value

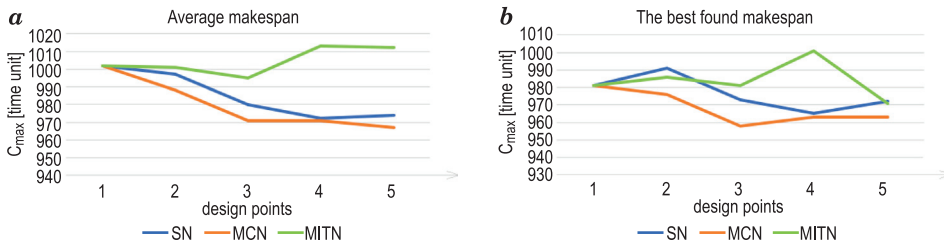

Fig. 4. Influence of other combinations (SN, MCN, and MITN values) on a – the average makespan and b – best makespan (C_{max} value)

for FT10 equals 930), a tenfold increase of the MCN value produced a result better by 9 time units (from 980 to 971), which is an almost 1% improvement compared to the optimal value.

In the experiment, the values of control parameters were adjusted for each problem according with formulas give in (KARABOGA, BASTURK 2007), but "number of operations" is used instead of fixed value "10": SN = $5 \cdot n$; MCN = $n \cdot m \cdot o$; and MITN = $2 \cdot n \cdot m$; where: n – number of jobs, m – number of machines, o – number of operations for a job. The proposed algorithm has been tested on 10 job shop scheduling benchmark problems (with 10 iterations) and the outcomes are presented in Table 3.

Table 3

Performance of DABC algorithm (C_{max} value)

Problem size	Performance of DABC algorithm (C_{max} value)			
	optimal makespan BKS	average found makespan	optimal makespan for DABC	RPI [%]
FT6	55	55	55	0.0
FT10	930	1,005	967	8.0
FT20	1,165	1,275	1,216	10.0
La02	655	676	655	3.0
La19	842	885	863	5.0
La21	1,046	1,152	1,102	10.0
La27	1,235	1,402	1,318	14.0
La30	1,355	1,461	1,404	9.0
La40	1,222	1,367	1,345	12.0
SWV11	2,983	3,944	3,844	32.0

Results comparison for JSSP test problems

KLOUD and KOBLASA (2011) compare SPT, FIFO, and GA taking into account C_{max} values and computational time. Only C_{max} values are compared in this work because of hardware differences. The comparison results are given in Table 4 (C_{max} and RPI values) for GA (KLOUD, KOBLASA 2011), PSO (GRACZYK 2017), DABC (WITKOWSKI et al. 2016), and Teaching-learning-based optimization (WITKOWSKI et al. 2016).

Table 4

C_{max} and RPI values for other algorithms

Problem size	Algorithms					
	GA	PSO	DABC	TLBO	HPSO	DPSO
FT6	57(4.0)	57 (4.0)	55 (0.0)	55 (0.0)	–	–
FT10	974 (5.0)	1,055 (13)	967 (4.0)	980 (5.0)	930 (0.0)	938 (0.8)
FT20	1,198 (3.0)	1,170 (0.5)	1,216 (4.0)	1,225 (5.0)	–	–
La02	668 (2.0)	664 (1.0)	655 (0.0)	686 (5.0)	655 (0.0)	655 (0.0)
La19	876 (4.0)	922 (9.0)	863 (2.0)	894 (6.0)	842 (0.0)	842 (0.0)
La21	1,098 (5.0)	1,218 (16)	1,102 (5.0)	1,145 (9.0)	1,078 (2.0)	1,047 (0.1)
La27	1,350 (9.0)	1,490 (20)	1,318 (7.0)	1,373 (11)	1,257 (3.0)	1,236 (0.1)
La30	1,362 (1.0)	1,567 (15)	1,404 (4.0)	1,411 (4.0)	–	1,355 (0.0)
La40	1,289 (5.0)	1,401(14)	1,345 (10)	1,326 (9.0)	1,224 (0.1)	1,229 (0.6)
SWV11	3,330 (12.0)	3,589 (20)	3,844 (29)	3,472 (16)	–	–

For each algorithm, we use formula RD = 100 (MFM-BKS)/BKS for each instance to calculate the relative deviation RD, where MFM means the minimum C_{max} found and BKS means the best known solution. We use ARD to denote the average value of relative deviations for all the analyzed instances.

For small problems FT06, FT10, FT20 and La02, almost all the algorithms can find good solutions. DABC algorithm shows better C_{max} values than TLBO for GA and PSO with FT10, La02, La19 and La27 problems. PSO algorithm shows better C_{max} values than DABC only for FT20 and SWV11 problems. TLBO algorithm shows better C_{max} values than DABC and PSO and worse than GA for La 40 and SWV11 problems, and GA works very well for solving La30, La40 and SWV11 problems.

Contemporary research on the use of PSO and ABC for solving JSSP problems includes various types of hybrid algorithms resulting from the combination of PSO and ABC with other algorithms. Examples of such applications are SONG (2008) and RAMESHKUMAR and RAJENDRAN (2018), which were used to compare the efficiency of the algorithms. SONG (2008) proposes an idea of a hybrid optimization algorithm HPSO. In order to prevent the algorithm falling

in a local optimization too early, Taboo Search is adopted to realize local parallel search, simultaneously improving the local search ability. In RAMESHKUMAR, RAJENDRAN (2018) the bench-mark JSSP problems are solved by discrete version of the PSO algorithm. DPSO algorithm show the best-known solutions to 3 out of 7 problems (La02, La19 and La30) in this work (Tab. 4). Figure 5 present C_{max} values for job shop scheduling test problems.

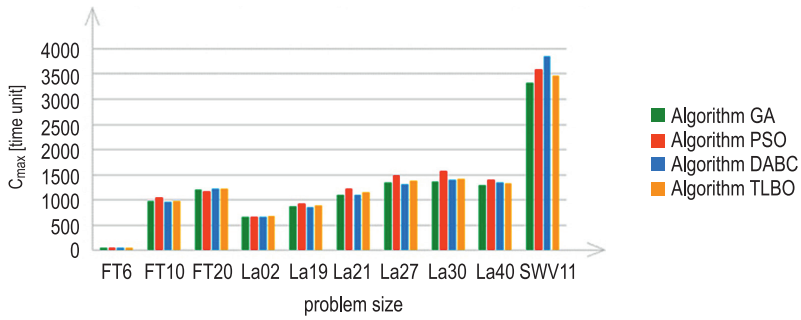

Fig. 5. $C_{max.}$ values for job shop scheduling test problems

A computational experiment with PSO, DABC and other algorithms for a real system

For a real production system the input data include: the matrix of groups of technologically interchangeable machines, the matrix of technological routes, the matrix of operations with an accuracy of group of technologically interchangeable machines, the matrix of processing times t_{ij} i-th of an operation, the matrix of processing times of a setup of machines before proceeding the j-th operation and i-th part. The data set contains 10 parts which need to be processed by 27 machines and 160 operations. The objective is to minimize C_{max} value for the FJSSP problem with a serial type production flow. In a serial production flow an entire batch of parts is processed on one machine and only when all of the parts in the batch have been processed are they sent to the next machine. Table 4 present a comparison of the results (C_{max} value) achieved by the different algorithms under discussion in the two versions of the problem, i.e., the serial flow and the serial-parallel flow (only the GA algorithm). For the real job shop problem with a serial-parallel route, it is not possible to compute C_{max} value analytically.

Analyzing the effectiveness of algorithms is a difficult task. For example, in Table 5 we can see that for the FJSSP problem with serial production flow ANN (WITKOWSKI et al. 2007), PSO (GRACZYK 2017), TLBO and DABC (WITKOWSKI et al. 2016) algorithms give better C_{max} values than the genetic algorithm

(WITKOWSKI 2016). But the GA algorithms were not as thoroughly tested as was the case with PSO, DABC and TLBO. There are specific values for mutation and crossover probability as well as many different types of mutation operators, crossover operators and type of selections for which we can obtain best experiment results. The following are applied in GA: population number = 1,000, generation number = 50, single swap mutation, order-based crossover and roulette wheel selection.

Table 5

C_{max} values for real scheduling problem

Production flow	Algorithms				
	GA	ANN	PSO	DABC	Optimal C_{max} value
Serial flow	57,636.0	50,242.2*	50,242.2*	50,242.2*	50,242.2
Serial-parallel flow	32,084.0	–	–	–	?

Conclusions

This work examined the JSSP problems with the objective of minimizing makespan. Computational results for JSSP problems were compared with some algorithms such as: GA, TLBO (non-hybrid algorithms) and HPSO, DPSO (hybrid algorithms). According to the theorem of No Free Lunch, no intelligent optimization algorithm is better than other intelligent algorithms. That is, every algorithm has its corresponding application circumstances. For small problems (FT6-La21) almost all the algorithms can find good solutions for JSSP problems. For relatively large problems (La30-SWV11), the results of the proposed algorithms DABC and PSO are worse than GA. The computational experiment shows that results given by the DABC and PSO for real job shop scheduling problem generated optimal C_{max} values. In future, we can expect more research to be done on the serial-parallel production flow, where individual items in a batch are sent to the machines as soon as they have been processed on the previous machine.

References

ABU-SRHALM A., AL-HASAN M. 2015. *Hybrid Algorithm using Genetic Algorithm and Cuckoo Search Algorithm for Job Shop Scheduling Problem*. International Journal of Computer Science Issues, 12(2): 288-292.

ADAMS J., BALAS E., ZAWACK D. 1988. *The shifting bottleneck procedure for job shop scheduling*. Management Science, 34(3): 391–401.

Błażewicz J., Ecker K., Pesch E., Schmidt G., Węglarz J. 2007. *Handbook on Scheduling; From Theory to Application*. Springer, Berlin, Heidelberg, New York.

Fisher H., Thompson G. 1963. *Probabilistic Learning combinations of local job shop scheduling rules*. Englewood Cliffs, New York, Prentice-Hall.

Graczyk P. 2017. *Particle swarm optimization for job shop scheduling* (*master thesis*). Warsaw University of Technology, Warsaw.

Krause, J., Cordeiro, J., Parpinelli, R.S., Lopes H.S. 2013. *A Survey of Swarm Algorithms Applied to Discrete Optimization Problems*. In: *Swarm Intelligence and Bio-Inspired Computation*. Eds. X.-S. Yang, Z. Cui, R. Xiao, A.H. Gandomi, M. Karamanoglu. Elsevier Inc., p. 169-191

Karaboga D., Basturk B. 2007. *Artificial Bee Colony (ABC) Optimization Algorithm for Solving Constrained Optimization Problems*. In: *Lecture Notes in Artificial Intelligence*. Eds. P. Melin, O. Castillo, L.T. Aguilar, W. Pedrycz. Springer-Verlag, Berlin, Heidelberg, p. 789–798.

Kloud T., Koblasa F. 2011. *Solving job shop scheduling with the computer simulation*. The International Journal of Transport & Logistics, 3: 7-17.

Lawrence S. 1984. *Resource constrained project scheduling, An experimental investigation of heuristic scheduling techniques*. Technical Report, GSIA, Carnegie Mellon University.

Mesghouni K., Hammadi S., Borne P. 2004. *Evolutionary Algorithms for Job Shop Scheduling*. International Journal of Applied Mathematics and Computer Science, 14(1): 91–103.

Rameshkumar K., Rajendran C. 2018. *A novel discrete PSO algorithm for solving job shop scheduling problem to minimize makespan*. IOP Conference Series: Materials Science and Engineering, 310: 10.

Shua D.Y., Hsu Ch.Y. 2006. *A hybrid particle swarm optimization for job shop scheduling problems*. Computers & Industrial Engineering, 51: 791-808.

Song X., Yang C., Qiu-Hong M. 2008. *Study on particle swarm algorithm for Job Shop Scheduling Problems*. Systems Engineering and Electronics, 30(12): 2398-2401.

Storer R., Wu D., Vaccari R. 1992, *New Search Spaces for Sequencing Problems with Application to Job Shop Scheduling*. Management Science, 38: 1495-1509.

Surekha P., Sumathi S. 2010. *PSO and ACO based approach for solving combinatorial Fuzzy Job Shop Scheduling*. International Journal of Computer Technology and Applications, 2(1): 112-120.

Witkowski T., Strojny G., Antczak P. 2007. *The Application of Neural Networks for Flexible Job Shop Problem*. International Journal of Factory Automation, Robotics and Soft Computing, 2: 116-121.

Witkowski T., Antczak A., Antczak P. 2010. *Comparison of Optimality and Robustness between SA, TS and GRASP Metaheuristics in FJSP Problem*. Lecture Notes in Computer Science, 6215: 319-328.

Witkowski T. 2016. *Scheduling Algorithms for Flexible Job Shop Scheduling*. Wydawnictwo Naukowe PWN, Warszawa.

Witkowski T., Krzyżanowski P., Vasylishyna S. 2016. *Comparison of DABC and TLBO Metaheuristics for Solve Job Shop Scheduling Problem*. 12th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery, Changsha, China (poster).