ISSN 1330-3651
UDC/UDK 658.512/.514:510.64

ADAPTIVE FUZZY PARTICLE SWARM OPTIMIZATION FOR FLOW-SHOP SCHEDULING PROBLEM

Vjekoslav Galzina, Roberto Lujić, Tomislav Šarić

Preliminary notes

This paper describes the application of a hybrid of fuzzy logic and swarm intelligence in order to achieve sub-optimal solutions for flow-shop scheduling problem. A novel adaptive approach with fuzzy particle swarm optimization is proposed. The developed model is tested with the standardized test functions and compared with selected stochastic algorithms (first with one objective functions and later with multi objective functions) to determine its applicability to general problems. Benchmark examples were utilized to evaluate the approach and determine the optimal number of the algorithm evaluations. Finally, the proposed model is applied on two practical problems of flow production problems (assembly lines and packaging lines). The results achieved were compared with the conventional priority rules and the effectiveness of the application of hybrid fuzzy logic and adaptive particle swarm optimization algorithm was demonstrated.

Keywords: *fuzzy logic, particle swarm optimization, scheduling, flow-shop*

Primjena adaptivnih neizrazitih rojeva čestica u optimizaciji raspoređivanja protočne proizvodnje

Prethodno priopćenje

Ovaj rad razmatra novi pristup problemu raspoređivanja u protočnoj proizvodnji korištenjem kombinacije neizrazite logike i optimizacije rojevima čestica u cilju postizanja sub-optimalnog rješenja. Predlaže se upotreba Tip-1 i Tip-2 modela neizrazite logike u kombinaciji s adaptivnim modelom rojeva čestica. Razvijeni model je uspoređen na standardiziranim testnim funkcijama za stohastičke algoritme (prvo jednokriterijske, a zatim višekriterijske postavljene funkcije cilja) kako bi se utvrdila njegova upotrebljivost na opće postavljenim problemima. Zatim je testiran na standardiziranim testnim zadacima za probleme protočne proizvodnje te konačno na dva praktična problema protočne proizvodnje (linije montaže i linije pakiranja). Rezultati ostvareni novim modelom su uspoređeni s konvencionalnim pravilima prioriteta te je pokazan kvantitativan i kvalitativan napredak primjenom hibrida neizrazite logike i rojeva čestica.

Ključne riječi: *neizrazita logika, optimizacija rojevima čestica, raspoređivanje, protočna proizvodnja*

1 Introduction

Scheduling is the process of making decisions about allocating resource activities (or activities over resources) within the time period in order to optimize one or more goals called objective functions. The process of scheduling (sometimes also called operational planning) provides sub-optimal schedule of the resources in the available time. In engineering problems, scheduling can be classified as a multitude of everyday tasks ranging from planning, design of products or services, and the exploitation and maintenance of the systems, plants and equipment. Even when it comes to producing a simple product, plan or project plans variants that can be scheduled becomes almost uncountable. As it is shown in [1] there is a constant need of innovative approaches in every segment of manufacturing industries development and deployment. The problem is one of dynamic scheduling problems, as they often contain incomplete, vague, and sometimes false information, whose dynamics can be revealed through the incompleteness of the data, uncertainty data, unexpected and unforeseen changes, lack of accuracy and precision in the duration of individual activities and/or necessary resources; restrictions that may be too strict or insufficiently strictly appointed while at the same time can be variable in time. Scheduling objective function defines the optimal schedule and constraints define the feasibility of the schedule.

Scheduling problem in manufacturing in cases with more than two or three resources are proven nondeterministic polynomial, NP hard problems [2], where the introduction of more than one objective function further makes the problem even more complex. Contribution to the complexity of the problem is introduction of imprecision and inaccuracy that inevitably exist in the world that surrounds us.

Recent advances in solving theoretical and practical flow-shop scheduling problems show that beside conventional methods (mathematic modelling and mainly priority rules) different metaheuristic optimization methods and approaches are used: Differential Evolution, DE [3], Genetic Algorithm, GA [4, 5], Tabu Search, TS [6], Ant Colony Optimization, ACO [5, 7], Bee Colony Optimization, BCO [8], Simulated Annealing, SA [9], Particle Swarm Optimization, PSO [9, 10] and others. Previously mentioned methods are sometimes combined and also can be extended with soft computing techniques like fuzzy logic, FL [11], neural networks, software tools like ARENA [12] and other.

2 Flow-shop scheduling

According to [13] there are two dissimilar types of production based on the flow of activity: flow-shop and job-shop productions. In practice, job-shop corresponds to an individual production, while flow-shop corresponds to the serial production and the process industries. For a job-shop scheduling problem the order of resource use is not defined in advance and often not even the sequence of a particular resource. The flow-shop scheduling problem has a pre-defined sequence of resource use, and thus the sequence of execution of a process. Figure 1 shows the proper sequence of activities on resources for basic flow-shop scheduling problem. Beside these types, as a specific type of problems emerges the project scheduling problem, which is characterized by significant deviations due to the imperfections and incompleteness of input data – and can be considered a borderline case of the scheduling problem [14]. The reality of production scheduling is the dynamics of events that inevitably lead to re-scheduling if unexpected events occur, such as resource deficiencies (failures, unavailability, inadequacy ...), deviations of activities, changes in terms of delivery and related feasible appearance.

By flow-shop scheduling (FSS) in this paper the general production scheduling problem in which each of *n* tasks or activities goes through the same sequence of *m* resources will be discussed. Subtype FSS problem for such a defined sequence of production in which every activity must go through each resource is called a permutation scheduling problem.

Let there be *m* resources and *n* activities. The duration of each activity is defined as a p_{ij}, $1 \leq i \leq m$, $1 \leq j \leq n$, therefore defined by the order limit which says that for each activity $j = 1, 2, 3, ..., n$ should be sent to *resources*1, then *resources*2, then *resources*3 and so on. The problem is, therefore, to find a schedule of activities π_j for each resource j ($\pi = (\pi_1, \pi_2, \ldots, \pi_n)$) which will have a minimum total time, minimum flow time through a given schedule and a minimum number of delayed activity. To calculate the completion time of scheduling type $n \times m$ problem the following can be stated [2, 13, 14]:

$$C(\pi_j, i) = \max\left[C(\pi_{j\text{-}1}, i), C(\pi_{j\text{-}1}, i-1) + p_{\pi_j, i}\right], \quad (1)$$
for $j = 2,..,n; i = 2,\ldots,m.$

Total duration of activity at all *m* resources can be defined as:

$$C_{\max}(\pi) = C(\pi_n, m). \quad (2)$$

Total processing time to find such a schedule π^* in the set of all schedules Π is defined as:

$$C_{\max}(\pi^*) = C(\pi_n, m), \ \forall \pi \in \Pi. \quad (3)$$

If the delivery time is marked with d_j which must satisfy the specified *j*-th activity then the target schedule π^* with minimum delay objective function can be defined as follows:

$$T_{\max}(\pi^*) = \max\left[(C(\pi_j, m) - d(\pi_j)), 0\right], \ \forall \pi \in \Pi. \quad (4)$$

It is then necessary to find a schedule π^* from the set of all schedules Π that validates the inequality:

$$T_{\max}(\pi^*) \leq T_{\max}(\pi), \ \forall \pi \in \Pi. \quad (5)$$

3 Adaptive fuzzy particle swarm optimization

Fuzzy logic systems and evolutionary algorithms (particle swarm optimization as a type of swarm intelligence algorithm) can be considered as the two components of general soft computing and have nature inspired origins; both are used as an extension to the conventional methods for solving general and specific problems of design, control, modelling and optimization.

3.1 Particle swarm optimization

Particle swarm optimization, PSO, is a stochastic optimization algorithm based on the population of solutions. It is originally developed by J. Kennedy and R. C. Eberhart [15] and presented as a new optimization method inspired by the phenomenon of biological and social-cognitive behaviour of different types of group-living creatures. Living in groups helps individuals in self-realization by means of social influence and social learning. Interaction with others produces the change of beliefs, views, and ultimately of the behaviour of individuals. These changes in the limited socio-cognitive space are represented as the movement of individuals. In other words, self-knowledge and knowledge of the group is optimized by the social contact because each individual can contribute to the overall knowledge of the group, the group transfers its knowledge by means of social contact to the individual members of the group. Clearly, these mechanisms are simplified descriptions of complex natural phenomena, but can faithfully illustrate the movement of swarms in search of better solutions. PSO algorithm uses a population-based search in which the particles change their position (state) by means of iterative algorithms. During the performance of the algorithm the particles change their positions in a multidimensional search space, i.e. the particles move in solutions space in search for the best positions.

Basic PSO algorithm

A swarm of particles as it is defined in the PSO algorithm shows stochastic behaviour because it uses velocity to update the current position of each particle in the swarm. Velocity is updated based on the memory of each particle - corresponding to autobiographical memory and based on the knowledge acquired by the swarm as a whole - which corresponds to learning from others. Specifically, these two functions are defined over cognitive and social parameters of speed update. Cognitive parameter determines the magnitude of the particles' confidence in its previous decision. Social parameter determines the magnitude of the particles' confidence in the other particles of the swarm. For example, zero for no confidence, one for complete trust. Position of particles in the swarm is updated based on the social behaviour of a swarm that adapts to its environment constantly looking for better positions over time (e.g. iterations).

Computing the new position $x(t+1)$ *i*-th particle in the next iteration $k+1$ is defined as [15, 16]:

$$x_i(t+1) = x_i(t) + v_i(t+1). \quad (6)$$

That is, the new velocity of particle where the new speed or velocity of the particle $v_i(t+1)$ in the next moment or iteration $t+1$ is calculated as:

$$v_i(t+1) = wv_i(t) + c_1 r_1 (p_i(t) - x_i(t)) + c_2 r_2 (g(t) - x_i(t)) \quad (7)$$

with:
$v_i(t)$ - speed or velocity of the *i*-particles in the previous position or previous point,
$x_i(t)$ - position or solution of the *i*-particles in the previous iteration or the previous point,

$p_i(t)$ – the best immediate solution of the *i*-particles,
$g(t)$ – the current global best solution of all particles,
r_1, r_2 – randomly selected number from the interval of real numbers [0, 1],
C_1 – cognitive or self-awareness parameter (real number),
C_2 – social or social parameter (real number),
w – inertia parameter (real number).

Fig. 1 shows the changing position of the *i*-particles from the moment to the next moment $t + 1$. The picture shows the position of the particles and the update of the velocity vectors as previously described in two-dimensional space. The updated position of the particles will depend not only on the best positions and the swarm of particles, but also on the amount of inertial, cognitive and social parameters of the algorithm.

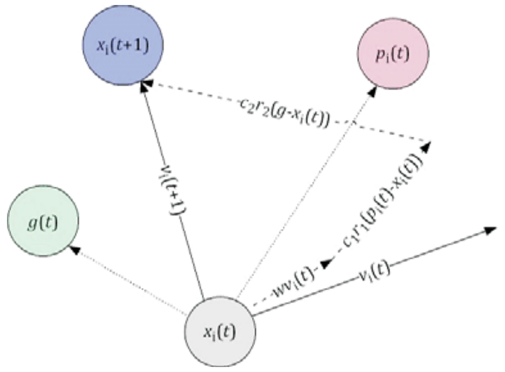

Figure 1 One step change of the position *x* for *i*-particle of PSO algorithm

Over time the different adaptations of the initial PSO algorithm were introduced [15, 16, 17] such as the addition of inertia, the speed limit, change the flow model of computation speed, the specific topology of local neighbourhoods particles (rings, stars, pyramids, Von Neumann's network model …) adoption of mechanisms from other evaluation algorithms (specific selections, crossing, mutation, tunnelling …). In this paper initial PSO algorithm is expanded with topology and particles interaction based on M. Clerc TRIBES algorithm [18]. The main alteration compared to the initial PSO algorithm is the usage of groups of particles called tribes (every tribe has its own leading particle) and additional selection and ranking of particles within tribes and beyond powered by fuzzy logic mechanisms implemented in algorithm.

3.2 Fuzzy logic

Fuzzy logic, FL is derived from the development of fuzzy sets by L. Zadeh (1965). Fuzzy logic in a broad sense serves as a tool for fuzzy control, analysis of vagueness in natural language and several other application domains. It is one of the techniques of soft-computing tolerant to impreciseness and incompleteness and giving back sufficiently good solutions [19]. The structure of conventional fuzzy system that is characterized by using type-1 fuzzy sets, which are defined on a universe of discourse, map an element of the universe of discourse onto a precise number in the real number unit interval [0, 1][11, 14, 19]. The concept of type-2 fuzzy sets was initially proposed by L. Zadeh (1975) as an extension of type-1 fuzzy sets. Higher types of fuzzy logic are not considered.

Interval type-2 fuzzy logic system, IT2 FLS according to J. M. Mendel and N. N. Karnik [20, 21] must contain at least one interval type-2 fuzzy set, IT2 FS, and consists of the following basic components (as illustrated in Fig. 2):

- Preparation of input data
- Fuzzification of input values (at least one must be IT2 FS type)
- Fuzzy rules database
- Evaluation of input and output fuzzy values by means of fuzzy rules database
- Reduction of type
- Defuzzification of output variables.

Layout like that the system can be seen as a way of mapping the real input values in the real output values. The input variables can be accurate and precise (perfect measurement, exact duration, etc.) and inaccurate (measured with noise, the estimated duration, etc.).

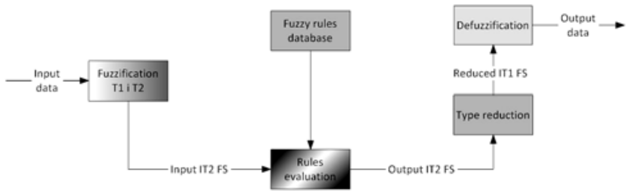

Figure 2 General schema of IT2 FLS

In this work, beside particle selection and ranking, the fuzzy logic is used for the exploitation and the determination of goodness, i.e. improvements that are achieved according to objective function.

According to the authors [20] IT2 FS can be used successfully in cases where T1 FS cannot adequately represent variables noting that there is currently no theoretical evidence to show that the IT2 FS is better than the T1 FS [19] but there is evidence of better performance. IT2 FS is reduced down to T1 FS by means of centre of sets type reducer and implemented as Karnik-Mendel algorithm as shown in [19, 21].

3.3 Pareto optimization

A popular approach to solving problems of multi-objective optimization, MO is based on the Pareto principle - named after the industrialist and economist V. Pareto (1848.-1923.). The Pareto optimization says that the MO solution to the problem is Pareto optimal (sometimes also called efficient, non-inferior or non-dominated) if the value of any objective function cannot be improved without degrading at least one other objective function of a solution vector. Two Pareto optimal solutions cannot be ranked or compared without some sort of additional set of criteria [9, 22]. General steps for finding the Pareto optimal solutions are: finding a set of Pareto solutions, or a representative subset of Pareto solutions, and then selecting one Pareto optimal solution according to the criteria set out. The set of all Pareto solutions is called Pareto front.

MO problem is defined as minimizing or maximizing the objective function with multiple constraints of inequality and equality such that according to [9, 11, 22-24]:

$$\min f(x) = (f_1(x), f_2(x), f_3(x), \ldots, f_N(x)) \tag{8}$$

$$g_k(x) \geq 0, \quad k = 1, 2, 3, \ldots, K \tag{9}$$

$$h_l(x) = 0,\ l = 1, 2, 3, \ldots, L \qquad (10)$$

with:
N – number of objective functions,
g – limit of inequality, $k = 1, 2, 3, \ldots, K$,
K – number of inequality constraints,
h – equality constraints, $l = 1, 2, 3, \ldots, L$,
L – number of equality constraints.

Fig. 3 shows an example of Pareto front with the minimization of two functions (f_1 and f_2). The squares in the figure represent possible solutions (solutions below the Pareto fronts are not possible because of the set limits and direction of the front) and the circles represent efficient solutions (the set of all Pareto efficient solutions makes the front). Point C is dominated by a solution that is less efficient than solutions a and b, which lie on the Pareto front.

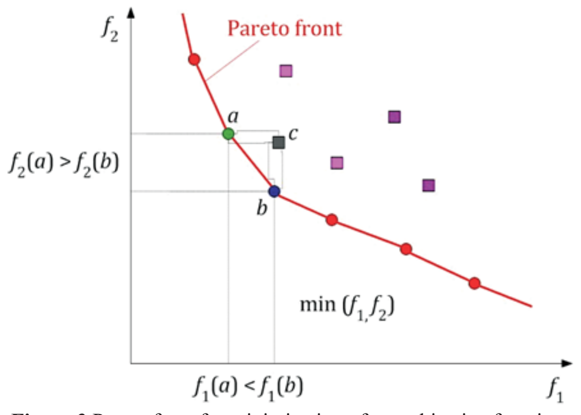

Figure 3 Pareto front for minimization of two objective functions f_1 and f_2

One of the main reasons why the methods of evolutionary algorithms are used in this type of problems is their ability to search the global solution space that can be discontinuous or have concave segments. These types of multi-objective optimization problems are difficult or irresolvable by means of traditional methods such as gradient or simplex methods [18, 23].

If n of the activities is given and must be performed on each of the m resources. And let p_{ij}, $i = 1, 2, \ldots, m$, $j = 1, 2, \ldots, n$, be the defined time it takes the j-th activity to finish on the resource i, and t_{ij}, $i = 1, 2, \ldots, m$, $j = 1, 2, \ldots, n$ start time of j activity on the resource i. Default set of flow production schedule is thus: $(1)_j$, $(2)_j$, ..., $(m)_j$. Also, let the default delivery date of the j-th activity be identified as d_j, $j = 1, 2, \ldots, n$,. The problem is the determination of sub-optimal sequence of activities to schedule resources for π. The objective function can be then defined as [21]:

$$F(\pi) = \min\left\{C_{\max}, \sum C_j, T_{\max}\right\}. \qquad (11)$$

Two schedules and π_2 are compared to the Pareto principle in order to comply with the following inequalities:

$$\begin{aligned}&\min C_{\max}(\pi_1) \le \min C_{\max}(\pi_2) \text{ and}\\&\min \sum C_j(\pi_1) \le \min \sum C_j(\pi_2) \text{ and}\\&\min T_{\max}(\pi_1) \le \min T_{\max}(\pi_2)\end{aligned} \qquad (12)$$

$$\begin{aligned}&\min C_{\max}(\pi_1) < \min C_{\max}(\pi_2) \text{ or}\\&\min \sum C_j(\pi_1) < \min \sum C_j(\pi_2) \text{ or}\\&\min T_{\max}(\pi_1) < \min T_{\max}(\pi_2).\end{aligned} \qquad (13)$$

If all of the conditions in Fig.12 and at least one of the conditions in Fig.13 are satisfied then the schedule π_1 is more efficient than π_2. Additionally, if the observed schedule is more efficient than all currently archived solutions then it enters the archive of optimal solutions.

Terms of the final selections have been placed through IT2 FLS that according to user settings, input the objective function and the set of fuzzy rule base selects a compromise solution from the archive solution - since it is not about to compare the various solutions in terms of schedule, but only in terms of achievement of the objective function (it is possible that there is more than one solution with a different order and the same values of the objective function in the archive of optimal solutions) [23].

4 Comparison results

Combination of soft-computing methods (PSO and FL) with the Pareto optimization approach is thus evaluated in comparison results: first for standardized test optimization problems [22, 23, 24], second with conventional scheduling priority rules on two strait-forward problems (Example 1 – scheduling of assembly line, Example 2 – scheduling of packaging line). Results for the spherical and fractal problem are presented first where spherical or De Jong is simpler; continuous, convex and unimodal and fractal (Weierstrass-Mandelbrot fractal, WMF) is a more complex problem.

4.1 Comparison results for common optimization problems

Fig. 4 shows that all algorithms (GA, SA, PSO, TRIBES and new fuzzy adaptive PSO, FA-PSO) had comparable results (test setup: 25 start-ups with 2000 iterations each; problem setup: 10 dimensions, precision of 0,01 and size of 100 units, [23]) and that swarm intelligence based algorithms (PSO, TRIBES and FA-PSO) scored somewhat worse than SA and GA on De Jong problem. And in Fig. 5 it is shown that FA-PSO scored best results compared with other swarm intelligence based algorithms on WMF problem (test setup: 25 start-ups with 2000 iterations each; problem setup: 10 dimensions, [23]).

The standard deviation of the results on De Jong and WMF problems for evaluated algorithms and all the results according to Ref. [23] are shown in Tab. 1. FA-PSO has shown consistent results on both problems and in range compared with TRIBES algorithm.

Table 1 Standard deviation of results for De Jong and WMF problem after 25×2000 iterations

	GA	SA	PSO	TRIBES	FA-PSO
De Jong	18,4381	112,9919	8,8107	**0,0163**	0,1741
WMF	0,2675	0,6763	0,4467	0,1598	**0,1264**

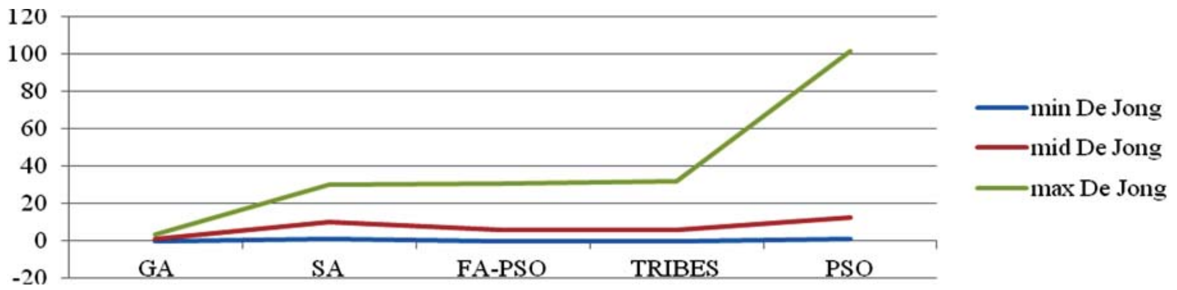

Figure 4 Comparison of minimal, medium and maximum results for De Jong problem after 25×2000 iterations (GA, SA, PSO, TRIBES and FA-PSO)

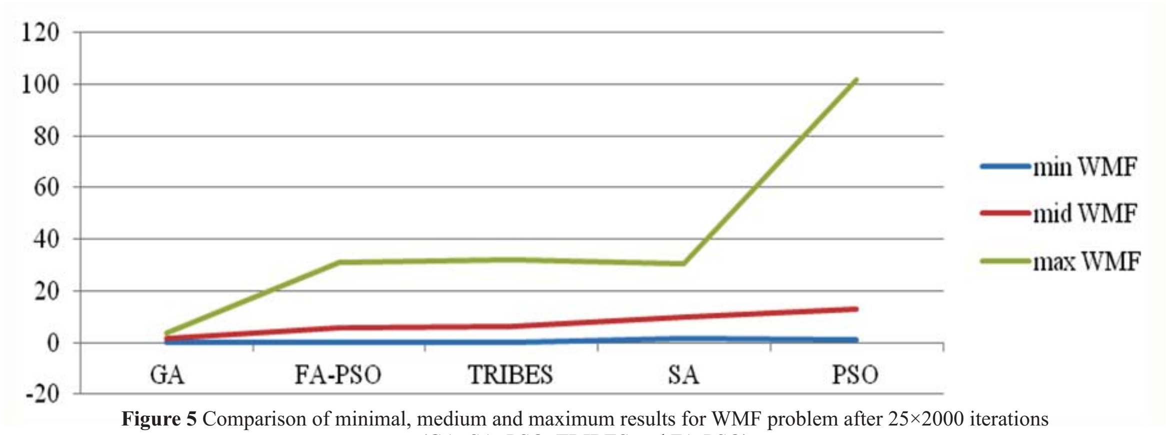

Figure 5 Comparison of minimal, medium and maximum results for WMF problem after 25×2000 iterations (GA, SA, PSO, TRIBES and FA-PSO)

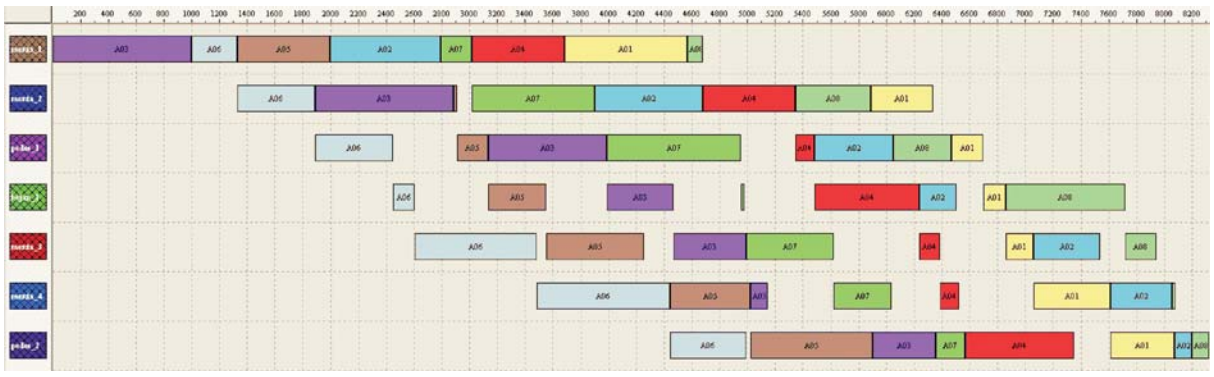

Figure 6 The resulting schedule of Example 1 using priority rule LPT

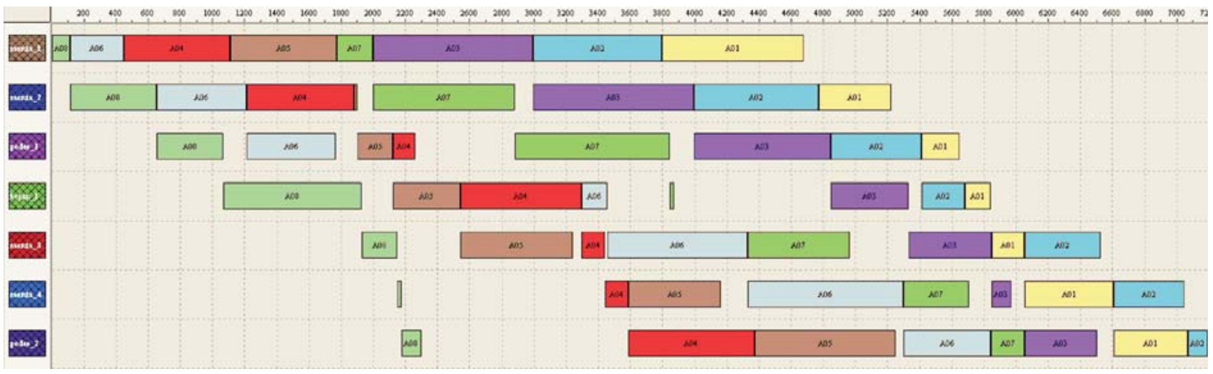

Figure 7 The resulting sub-optimal schedule of Example 1 using FA-PSO algorithm

4.2 Comparison results for flow-shop scheduling problems

Scheduling priority rules as expression of the relative importance of activity and thus affects the waiting time for the execution of the activities, the flow of time and overall end time [2, 13]. In assigning priorities to define the characteristics of activities on the basis of which is determined by the order of commitment and resources to each activity determine the appropriate relative importance. Limited number of priority rules [2, 14, 23] was used for comparison with the FA-PSO and included the following rules: First Comes First Served, FCFS, Last Come First Served, LCFS, Earliest Due Date, EDD, Shortest Processing Time, SPT, Longest Processing Time, LPT, Critical Ratio, CR. Example of solution for 7×8 flow-shop

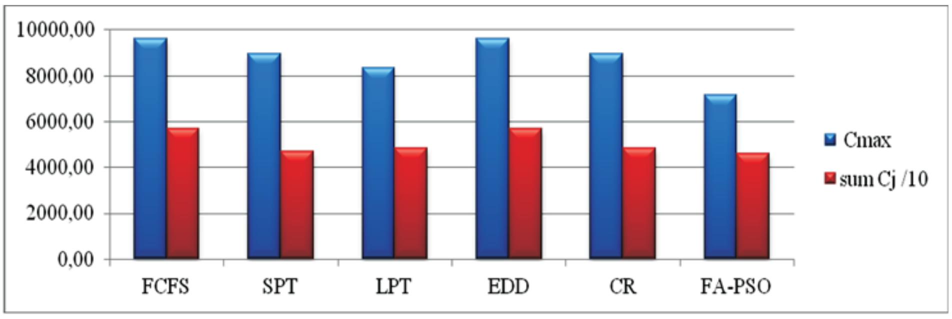

Figure 8 Comparison of completion time ($C_{\max}$) and total completion time (ΣC_j scaled by factor 10) for Example 1 (FCFS, SPT, LPT, EDD, CR and FA-PSO)

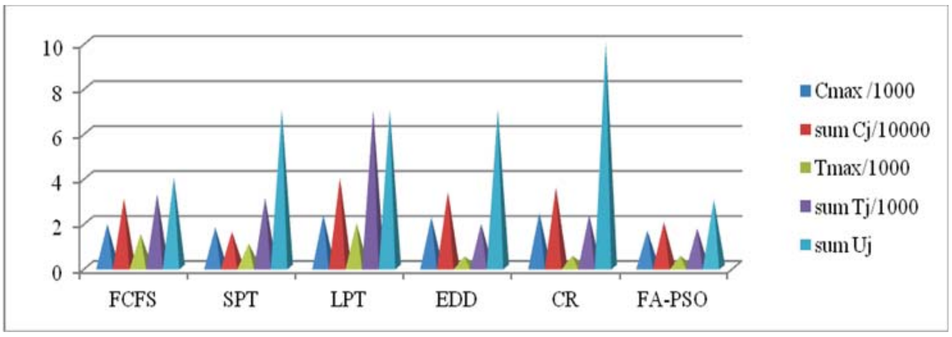

Figure 9 Comparison of overall results for Example 2 schedule optimization (FCFS, SPT, LPT, SPT, EDD, CR and FA-PSO)

scheduling problem of assembly line [23] with priority rule LPT used is shown in Fig. 6, and for FA-PSO in Fig. 7.

From the data visible in Fig. 8 can be said that the FA-PSO achieved favourable variant of schedule on designated Example 1 because the obtained values of the total duration of the overall schedule and the days required for completion of all activities schedule lower than is contemplated by the priority rules.

Furthermore, if we look at the comparison in Fig. 9 for Example 2, except for the above mentioned criteria and the criteria of tardiness and total overall tardiness and the number of late activities again FA-PSO achieved overall favourable variant of the schedule ever any of the observed priority rules.

5 Conclusion

Scheduling model based on FL and adaptive variant PSO for flow-shop problem is presented; FA-PSO is compared with GA, SA, native PSO and TRIBES algorithms and scored competitive results on common optimization problems and good results in standard deviation. Finally the model was applied on two problems and the obtained results were then compared with five conventional scheduling priority rules (FCFS, EDD, SPT, LPT, CR). In both instances the proposed FA-PSO algorithm showed better performance in all but speed of execution segment. It can be said that the FA-PSO, on average, is more successful compared to the conventional methods due to the built-in randomness that cannot be said for individual solutions which stands for this and any other stochastic algorithms. The proposed model is for now limited to the FSS problem with goal functions defined earlier. Advantages of the model are: smaller number/lack of parameters for optimization algorithm because of adaptive nature of the used PSO algorithm, no need of tuning-up optimization algorithm parameters, free user's definition of fuzzy variables and variables interaction. In this context fuzzification and defuzzification of values can also be a disadvantage because the definition of needed I/O variables enlarges the design process and makes it somewhat less straightforward to evaluate and compare final results.

Acknowledgements

This work is part of the research included in the project Number 152-1521781-2235 supported by the Ministry of Science, Education and Sports of the Republic of Croatia. The authors would like to thank the Ministry for the support of this work.

6 References

[1] Palcic, I.; Buchmeister, B.; Polajnar, A. Analysis of innovation concepts in Slovenian manufacturing companies. // Strojniški vestnik - Journal of Mechanical Engineering, 56, 12(2010), pp. 803–810.
[2] Bruckner, P. Scheduling Algorithms, 5th ed., Springer-Varlag Berlin Haidelberg, 2007.
[3] Udhayakumar, P.; Kumanan, S. Task Scheduling of AGV in FMS using non-traditional optimization techniques. // International Journal of Simulation Modelling, 9, 1(2010), pp. 28–39.
[4] Qian, B.; Wanga, L.; Hub, R.; Huang, D.X.; Wanga, X. A DE-based approach to no-wait flow-shop scheduling. // Computers & Industrial Engineering, 57, (2009), pp. 787–805.
[5] Pan, J.C.-H.; Huang, H.-C. A hybrid genetic algorithm for no-wait job shop scheduling problems // Expert Systems with Applications, 36, (2009), pp. 5800–5806.
[6] Grabowski, J.; Wodecki, M. A very fast tabu search algorithm for the permutation flow shop problem with makespan criterion. // Computers & Operations Research, 31, (2004), pp. 1891–1909.
[7] Berrichi, A.; Yalaoui, F.; Amodeo, L.; Mezghiche, M. Bi-Objective Ant Colony Optimization approach to optimize production and maintenance scheduling. // Computers & Operations Research, 37, 9(2010), pp. 1584-1596.

[8] Quan-Ke Pan, Q.K.; Fatih Tasgetiren, M.; Suganthan, P.N.; Chua T.J. A discrete artificial bee colony algorithm for the lot-streaming flow shop scheduling problem. // Information Sciences, 181, 12(2011), pp. 2455–2468.

[9] Varadharajan, T.K.; Rajendran, C. 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), pp. 772–795.

[10] Kuo, I. H.; Horng, S. J.; Kao, T. W.; Lin, T. L.; Lee, C. L.; Terano, P.; Pan, Y. An efficient flow-shop scheduling algorithm based on a hybrid particle swarm optimization model. // Expert Systems with Applications, 36, (2009), pp. 7027–7032.

[11] Lau, H. C. W.; Chan, T. M.; Tsui, W. T.; Chan, F. T. S.; Ho, G. T. S.; Choy, K. L. A fuzzy guided multi-objective evolutionary algorithm model for solving transportation problem. // Expert Systems with Applications, 36, (2009), pp. 8255–8268.

[12] Schroeder, R.G. Upravljanje proizvodnjom – odlučivanje u funkciji proizvodnje, četvrto izdanje, Mate d.o.o., Zagreb, 1999.

[13] Eberhart, R.C.; Kennedy, J. A New Optimizer Using Particles Swarm Theory. // Proc. Sixth International Symposium on Micro Machine and Human Science, Nagoya, Japan, 1995, pp. 39–43.

[14] Zadeh, L. Introduction // Fuzzy logic technology and applications / uredio Marks II, R. J. IEEE Publications, 1994.

[15] Engelbrecht, A. P. Computational intelligence: an introduction – 2nd ed., John Wiley & Sons Ltd, England, 2007.

[16] Floreano, D.; Mattiussi, C. Bio-Inspired Artificial Intelligence: Theories, Methods, and Technologies, MIT Press Cambridge, Massachusetts, 2008.

[17] Bonabeau, E.; Dorigo, M.; Theraulaz, G. Swarm intelligence: from natural to artificial intelligence, Oxford University Press, Inc., 1999.

[18] Clerc, M.; Cooren, Y.; Siarry, P. Performance evaluation of TRIBES, an adaptive particle swarm optimization algorithm. // Swarm Intelligence, 3, (2009), pp. 149–178.

[19] Mendel, J. M.; Hagras, H.; John, R. I. Standard Background Material about Interval Type-2 Fuzzy Logic Systems, Boilerplate Material, University of Essex, 2006.

[20] Castillo, O.; Melin, P. Studies in Fuzziness and Soft-Computing, Type-2 Fuzzy Logic: Theory and Applications, Springer-Verlag Berlin Heindenberg, 2008.

[21] Karnik, N. N.; Mendel, J. M. Centroid of a type-2 fuzzy set, Information Sciences, 132, (2001), pp. 195–220.

[22] Lu, H. State-of-the-art multiobjective evolutionary algorithms – Pareto ranking, density estimation and dynamic population, thesis, 2002.

[23] Galzina, V. Contribution to complex systems scheduling model by application of fuzzy swarm intelligence, thesis, University of Osijek, Mechanical Engineering Faculty in Slavonski Brod, 2011.

[24] Deb, K.; Thiele, L.; Laumanns, M.; Zitzler, E. Scalable multi-objective optimization test problems. // Proc. Congress of Evolutionary Computation, 1, 2002.

Authors' addresses

Vjekoslav Galzina
Strojarski fakultet u Slavonskom Brodu
Sveučilišta J. J. Strossmayera u Osijeku
Mechanical Engineering Faculty in Slavonski Brod
J. J. Strossmayer University of Osijek
Trg Ivane Brlić-Mažuranic 2
HR-35000 Slavonski Brod, Republic of Croatia
e-mail: vgalzina@sfsb.hr

Roberto Lujic
Strojarski fakultet u Slavonskom Brodu
Sveučilišta J. J. Strossmayera u Osijeku
Mechanical Engineering Faculty in Slavonski Brod
J. J. Strossmayer University of Osijek
Trg Ivane Brlić-Mažuranic 2
HR-35000 Slavonski Brod, Republic of Croatia
e-mail: rlujic@sfsb.hr

Tomislav Šaric
Strojarski fakultet u Slavonskom Brodu
Sveučilišta J. J. Strossmayera u Osijeku
Mechanical Engineering Faculty in Slavonski Brod
J. J. Strossmayer University of Osijek
Trg Ivane Brlić-Mažuranic 2
HR-35000 Slavonski Brod, Republic of Croatia
e-mail: tsaric@sfsb.hr