APEM journal
Advances in Production Engineering & Management
Volume 15 | Number 3 | September 2020 | pp 267–276
https://doi.org/10.14743/apem2020.3.364
ISSN 1854-6250
Journal home: apem-journal.org
Original scientific paper

Decentralized optimization of the flexible production lines

Malega, P.[a], **Rudy, V.**[a], **Kanász, R.**[b], **Gazda, V.**[c,*]

[a]Technical university of Kosice, Faculty of Mechanical Engineering; Institute of management, Industrial and Digital Engineering, Slovak Republic
[b]Marwin Cassovia Soft, Ltd.
[c]Technical university of Kosice, Faculty of Economics, Department of Finance, Slovak Republic

ABSTRACT

The shortening of the production cycle and increasing impact of the technological innovations evokes improvement of the methods used in the production line scheduling. The aim of the presented research is a proposal of the decision model that enables a flexible reaction to the changing production conditions. The central decisions are substituted by the decisions performed on the independent machines level. The machines utilize rather restricted information on the capacities utilization of their technological neighbours. The decisions follow the decision tables (decision chromosomes) re-coded in the course of the evolution. The Genetic algorithms standing behind the model, enable identification of the highly acceptable solutions that is proved by a set of the simulation experiments. The set of independent machines becomes self-organized, having a significant positive effect on the production line capacities utilization. The decentralization aspect makes our proposal somewhat different from other research in the field. The philosophy is built on the fact that each machine makes its production decision based on the available information, which it has at its disposal in a given time. The machine flexible reacts to its neighbours' capacity utilization (machine before and after given production machine) in the production process flow.

ARTICLE INFO

Keywords:
Production line;
Job shop problem (JSP);
Decentralised optimization;
Production scheduling;
Shortest processing time rule;
Self-organization;
Genetic algorithm;
Decision table

**Corresponding author:*
vladimir.gazda@tuke.sk
(Gazda, V.)

Article history:
Received 15 May 2020
Revised 16 October 2020
Accepted 19 October 2020

1. Introduction

Since the first mathematical formulations of production scheduling in the late 1950s, the topic became a subject of intensive scientific discourse. The problem formulations involve the assignment of scarce resources, typically machines, to the competing tasks over time to optimize some characteristics of system performance. Proper production scheduling promises considerable economizing of the production line performance achieved without paying any additional costs.

Development of operational research methods and growth of computational capacities significantly modified the character of the problem settings, covering the formulation of the dynamic programming, integer programming, and various types of the heuristic algorithms until the end of the 90s (for more detailed preview see [1]). However, the rapid development of the technologies, increasing production complexity and shortening of the production cycles limit the potential of central control of the production lines. Designing flexible production lines capable of reflecting sudden unexpected changes in the environment becomes inevitable. In this framework, the theory of Artificial Intelligence [2] offers the methods feasible for utilization in the new approach to production scheduling. Notably, the methodology of Agent based modelling becomes quite popular. Agents constitute the independent decision units enjoying simple behaviour rules

and following their objectives. Their mutual interactions lead to the synchronization of the whole system, often generating entirely unexpected macro-features – called emerging events. Workplaces (or machines)[1], playing the role of the agents, could be successfully utilized in the proposals of the new flexible production line scheduling. We assume that the self-organization of the workplaces enables a flexible reaction to the unexpected production shocks and quite stable solutions with regard to the changing environment.

The paper presents the application of the job shop scheduling based on the autonomous working place decisions utilizing the evolutionary mechanism of genetic algorithms. We show that even simple decision rules generated in the process of the genetic evolution lead to systematic shortening of the jobs makespan and thus economizing of the production lines. The simulation results are compared with the traditional approach to scheduling – in our case, the shortest processing time (SPT) rule. We show that the proposed approach provides computationally stable solutions.

The paper is organized as follows. The second section describes the literature motivating the presented research. The proposed model is introduced in the third section. The fourth section presents the simulation results, and the fifth section discusses the contributions of the paper.

2. Literature review

Different types of scheduling problems vary according to their organization, range, and other details. One of the most discussed types is the Job-shop problem (JSP), where a set of machines processes a group of jobs. Every job consists of a sequence of consecutive operations (tasks), while particular machines perform them in a reserved way. The scheduling constitutes in the assignment of the operations on the machines by optimizing a specific indicator [3]. The most traditional optimization criterion is a makespan, which is the time needed for processing all jobs in a given production schedule.

Besides the minimization of the makespan, the classical JSP also targets improving production efficiency, reduction of the production costs [4], or reduction of the energy consumption [5]. At the same time, it should take into consideration all the constraints known in advance [6]. When speaking about methods, mathematical models and optimizing algorithms dominated in the research published in the last decades of the 20th century [1].

The job-shop scheduling problem (JSP) belongs by its nature to the combinatorial optimization problems [7]. Garey *et al*. and Blazewic *et al*. pointed to its computational complexity that is NP-hard [8, 9]. Besides, most production systems operate in a dynamic environment where a few unpredictable real-time events can entirely disrupt the previously defined planned schedule. Examples of such real-time events include machine failures, the arrival of imperative jobs, due date changes, and others [10].

Mohapatra *et al*. [11] used a multi-objective approach to solve the planning and scheduling problem. Three different objectives considered in this work are the following: minimisation of (i) makespan, (ii) machining costs, and (iii) idle time of machines. To solve this integration problem, the authors propose an improved controlled elitist non-dominated sorting genetic algorithm to take into account the computational intractability of the problem.

Supsomboon and Vajasuvimon [12] proposed the alternative solutions to improve machinery production process in job shop scheduling using alternative strategies. The authors apply basic principles of operations management. Simulation models showed the great performance to help in making a decision in their real system.

Ma *et al*. [13] developed a novel scheduling method based on the fuzzy satisfaction rate and differential evolution algorithm. In the method, at first the fuzzy parameters are deduced based on normal distribution to calculate the satisfaction rate. After that a differential evolution is proposed.

[1] Terms workplace and machine are used as synonyms in the text

Ojstersek *et al*. [14] emphasises the advantage of a modular adaptive design allowing adaptation of the simulation model to a wide range of multi-objective optimisation problems. The presented MOHKA method is suitable for optimisation results' interactivity between the mathematical and simulation models. The interactive method pointed on the advantage of transferring the optimisation results via a simulation model to a real environment.

Shen and Yao's [15] contribution is the construction of a dynamic multi-objective optimization model for multi-objective dynamic flexible job shop scheduling. Another contribution of this research is the design of a dynamic decision making procedure.

On the other hand, neither the mathematical programming in its classical form nor the intuitive heuristics are capable of tackling the high complexity problems. On the other hand, at the end of the 20th and start of the 21st century, many new computational approaches emerged.

Meolic and Brezocnik [16] propose a novel method for generating and counting solutions of a flexible job shop scheduling problem. They represent the feasible solutions as cubes of a combination set. The algorithm is implemented as a sequence of operations in unique cube s*et al*gebra.

Yu *et al*. [17] propose a novel two-phase integration of process planning and scheduling approach. The preplanning phase generates a process network for each job with consideration of the static shop floor status. After achieving this goal, the final planning phase simultaneously creates the process plan of each job and the scheduling plan according to the current shop floor status.

The one, dominating the presented research, is the genetic algorithm (GA) method first formulated by J. Holland in his seminal works [18, 19]. However, GA applications in JSP scheduling has a long history starting with Bierswirth *et al*. [20] who published seemingly the firs**t**, however, mostly neglected, work.

Janes *et al*. [21] present an efficient genetic algorithm for solving job-shop scheduling problems. In comparison with traditional genetic algorithm, selection and crossover operators were modified. Authors demonstrate the viability of the proposed method for real-world problems and suggest its readiness for application in industry.

Chen and Hao [22] apply the non-dominated sorting genetic algorithm in combination with the multi-objective optimization. An excellent review of the scheduling under Industry 4.0 is provided by Zhang *et al*. [23].

In production scheduling, genetic algorithms (GA) represent schedules as individuals or a population's members. Every individual has its own fitness value. It is measured based on the objective function. This methodical approach works iteratively, and this iteration is a generation. Every generation has individuals and they are from the previous generations. Typically, the population size stays consistent starting with one generation to the next generation.

3. Proposed model

Job Shop Problem (JSP) is very famous optimization problem in software engineering and operational research. The focus is on assigning jobs to resources at particular times. A Job Shop is a work area where various universally useful workstations exist and are utilized to carry out a variety of jobs.

The proposed model assumes that the complexity of the production line prevents its effective centralized scheduling. Instead, the autonomous decision of the individual workplaces (machines) equipped with artificial intelligence is assumed. Similar to some bioinspired algorithms (flocking is the best example), the individual machines react to their neighbours to coordinate their job selection decisions. Thus the production line becomes self-organized and decentralised demonstrating a rather high level of synchronization.

3.1 Basic concepts and definitions

We assume a production line given as a set $M = \{1, 2, \ldots, m\}$ of the dedicated machines ordered according to technology process. If $i < k$ $(i, k \in M)$ then machine i precedes machine j in the technological flow. There are no machines running in parallel. Set $J = \{1, 2, \ldots, n\}$ denotes the

scheduled jobs (i.e., products). Function $t : J \times M \rightarrow N_+$ assigns task processing time $t_{i,j}$ of the i-th job to the machine j.

Operational imbalances of the time schedules imply the accumulation of the semi-products in each machine buffer store. It is fulfilled with the semi-products waiting for the follow-up of the production process. The machine decision consists of the selection and processing of one particular product among the stored semi-products. The machine interrupts production if it has no products in its buffer store. Any interruption prolongs the makespan of the whole production line.

Let us explain the situation of the i-th machine in detail. The i-th machine is preceded by the machine $i-1$ and succeeded by the machine $i+1$ in the technological flow. Because of the expression simplicity, the machine indexes are further substituted by −1, 0, +1. Machines do not dispose of the complete production line information and know just the content of the buffer store of both its own and the ones of −1 and +1 neighbours. The agent does not take into consideration the information regarding tasks, which have already been performed.

The useful information of the buffer store contents (see Eq. 1) is formally expressed as an array of the task capacity demands as follows

$$A^{(k)} = \left[a_{i,j}^{(k)}\right] \quad i \in J; \quad j,k \in \{-1,0,1\}; \quad j \geq k \tag{1}$$

where (k) denotes the buffer store of the k-th machine (predecessor (-1), current machine (0), and its successor (1)); i denotes i-th job; ordered pair (i,j) denotes the task, i.e. the i-th job time capacity requirement against the j-th machine.

An example of the information disposable for one machine is given in Table 1.

Each machine makes its independent decision on which product from its buffer store to choose for further processing. In such a decision situation, the traditional views on the production scheduling would offer mostly various variants of the Shortest Processing Time (SPT) heuristic. It assumes that if the machine is free, then it chooses the task with the smallest processing time requirements on the current machine. However, there are many objections to the effectiveness of this heuristic [24].

Table 1 Example of the buffer stores – arrays $A^{(-1)}$, $A^{(0)}$, $A^{(1)}$

	$A^{(-1)}$				$A^{(0)}$			$A^{(1)}$
Job (i)	$A_{i,-1}^{(-1)}$	$A_{i,0}^{(-1)}$	$A_{i,1}^{(-1)}$	Job (i)	$A_{i,0}^{(0)}$	$A_{i,1}^{(0)}$	Job (i)	$A_{i,1}^{(1)}$
1	7	6	8	5	1	2	8	5
3	5	3	2	6	3	1	9	2
2	1	7	1	4	8	4		
				7	6	6		

3.2 Genetic algorithm in strategy decision

The current state of artificial intelligence development offers a wide variety of inspirations for the SPT rule extensions. Highly promising seems to be a J. Koza's ([25]) theory of the genetic programming that is capable of deriving productive decision trees to solve rather complicated and in-formalized decision situations. It also remains the main inspiration for the proposed model. On the other hand, the substitution of the decision trees by rather simple decision tables significantly clarifies the decision logic and reduces the computational capacities requirements. Besides, we consider the deep decision trees to be less stable in the decision rules evolution.

Introduce the row vector-chromosome of the production line (see Eq. 2)

$$x = [x^1|x^2|\ldots|x^m] \tag{2}$$

that constitutes from the individual machine row vector-chromosomes. The chromosome of the ith machine (see Eq. 3) is defined as

$$x^i = \left[x_1^i, x_2^i, \ldots, x_p^i, x_{p+1}^i, \ldots, x_r^i,\right] \tag{3}$$

where its first p elements encode the simple logical conditions derived from the disposable information (i.e. $A^{(-1)}$, $A^{(0)}$, $A^{(1)}$ arrays). The last $r - p$ vector elements encode the product conditional selection strategy based on fulfilment given logical conditions.

All the vector chromosome elements (*genes*) are represented by the trits, which are 3-state analogues to the 2-state bits. Each gene contains one of −1, 0, 1 value (*allele* in genetics). The decision table with varying r bits enables modelling 3^r logical condition variants (see Table 2).

First $r - p$ trits have the following interpretation:

- if the gene gets number 1, then the corresponding logical condition is fulfilled;
- if the gene gets value -1, then the negation of the corresponding logical condition is fulfilled;
- if the gene gets value 0, the validity of the logical condition is inconclusive and is not taken into consideration.

Table 2 The one-machine chromosome vector composition – Example

Trit (gene)	Logical condition	Indicator (allele)
1	$\sum_i a_{i,1}^{(1)} > \max_i a_{i,0}^{(0)}$	{-1,0,1}
2	$\sum_i a_{i,0}^{(0)} > \max_i a_{i,-1}^{(-1)}$	
3	$\min_i \left(a_{i,1}^{(1)}\right) > \min_i \left(a_{i,0}^{(0)}\right)$	
4	$\mathrm{med}_i \left(a_{i,1}^{(1)}\right) > \max_i \left(a_{i,1}^{(0)}\right)$	
5	$\min_i \left(a_{i,-1}^{(-1)}\right) < \min_i \left(a_{i,0}^{(0)}\right)$	
6	$\max_i \left(a_{i,-1}^{(-1)}\right) > \max_i \left(a_{i,0}^{(0)}\right)$	
7	$\sum_i a_{i,-1}^{(-1)} > \sum_i a_{i,0}^{(0)}$	
8	$\max_i \left(a_{i,0}^{(0)}\right) > \max_i \left(a_{i,1}^{(1)}\right)$	
9	$\mathrm{med}_i \left(a_{i,-1}^{(-1)}\right) > \max_i \left(a_{i,0}^{(0)}\right)$	
Trit	Conditional Strategy	Indicator (allele)

10	if (cond.(1.-9.) == Ind.(1.-9.)) then {
	if (Ind(10.) == 1) then
	{choose arg $\max_i$ $(a_{i,0}^{(0)}$}
	else
	{if (Ind(10.) == −1)
	then
	{choose arg $\min_i$ $(a_{i,0}^{(0)}$}
	else
	{evaluate condition Ind(11.)}
	}
	}
		{-1,0,1}

11	if (Ind(11.) == 1) then
	{choose arg $\max_i$ $(a_{i,0}^{(0)} + (a_{i,1}^{(1)}$}
	else
	{if (Indicator == −1)
	then
	{choose arg med_i $(a_{i,0}^{(0)})$}
	else
	{choose arg $\min_i$ $(a_{i,0}^{(0)} + (a_{i,1}^{(1)}$}
	}
		{-1,0,1}

Notes: Trit means three-state analogy to the double-state bit. The genes trits contents (alleles) are set in the course of the genetic evolution. First nine trits code confront the logical condition: fulfilled (1); fulfilled negation (−1); not in consideration (0). Trits 10 and 11 code the conditional strategy of the product selection.

In case of the remaining $r - p$ trits, the conditional selection of a strategy is based both on the $-1, 0, 1$ corresponding trit content, and logical fulfilment of the selected conditions from the first r ones.

We assume the strategy space should contain just a few strategies to keep the machine strategy simple and transparent. In an example, we assume the selection of the i^*-th product for further processing based on the following:

- $i^* = \arg\max_i \left(a_{i,0}^{(0)}\right)$
- $i^* = \arg\min_i \left(a_{i,0}^{(0)}\right)$
- $i^* = \arg\max_i \left(a_{i,0}^{(0)} + a_{i,1}^{(1)}\right)$
- $i^* = \arg\min_i \left(a_{i,0}^{(0)} + a_{i,1}^{(1)}\right)$
- $i^* = \arg\text{med}_i \left(a_{i,0}^{(0)} + a_{i,1}^{(1)}\right)$

An example of the decision table with $p = 9$ and $r - p = 2$ elements is given in Table 2. In the following, the simulation results section provides an analysis and simulation results of the given example.

4. Simulation results and discussion

Genetic algorithms simulate the Darwinian and Mendelian principles of the evolution that are based on the natural selection of the most fitted individuals and their ability to accommodate to the changing competitive environment. The most fitted individuals (elite) are selected for breeding transferring their genetic information encoded in chromosomes onto the next generation. The rest of the population is assumed to die off and being replaced by the offspring of the breeding parents. The inheritance process is based on the crossing and mutation. While crossing directly transfers the chromosome crossing combination (inheritance information) from the parents onto the next population, the mutation introduces to the population a new piece of information, regardless of whether positive or negative. It has been proved that large and most heterogeneous populations provide best fitted individuals both in case of nature as well as in the case of the artificial simulations.

Table 3 Parameters settings and their interpretation

Parameter	Values	Interpretation
No of machines	10	Dedicated machines ordered according to the technological flow. No machine works in parallel.
No of jobs	30	
TPT	Unif (1,20)	Task processing times (TPTs) are uniformly distributed on the {1,2,…,20} set.
Prob. of crossing	0.8	When a pair of the machines is selected from the elite category, probability of their breeding is 0.8.
Population	100, 300, 500	Population consists of the set of the virtual production lines each endowed with its specific chromosome. Gene values (-1, 0, 1; alleles) are set randomly according to the uniform distribution in the start of the simulation.
Elitism	0.1,0.2,...0.9	The share of the most fitted individuals selected for breeding and surviving into the next simulation round.
Type of crossing	1-point	The parents' chromosomes cross in one randomly selected position (gene).
Mutation rate	0.01,0.05,0.1	Mutation is based on random flipping of the contents of one randomly selected gene (alelle).
End of simulations	50 periods	One period consists of consecutive steps: (1) determination of the elite; (2) crossing; (3) mutation of the offspring; (4) computation of fitness.
Fitness	Negative value of makespan	Fitness is given as a negative value of the makespan.

Simulation models are sensitive to the parameters setting. If set incorrectly, the simulation results usually fail. In case of the proposed model, a few pilot simulations were performed to identify an admissible parameters space worthy for the consideration as given in Table 3.

4.1 Optimal population size, mutation rate and elitism

Determination of the satisfactory population size seems to be crucial for identification of some kind of sub-optimal solution in an acceptable computing time. If one machine-chromosome contains 11 three-state genes (trits), there are $3^{11} = 177147$ possibilities how to code it. Complexity of the problem significantly arises, if considering the machine-line chromosome (if 10 machines, then 3^{110} permutations). In case the population is rather large and heterogeneous, it can contain enough information to achieve the optima by simple parents' chromosomes crossing. On the other hand, a small population may be rather depleted of the genetic content variability and the only chance to converge to some acceptable solutions is introducing of the new information via mutation, but lasting longer. A similar dilemma faces the proper determination of the elitism size. The small one bears the threat of losing the genetic variability causing convergence into a sub-optimal state located far from the optimal one. Alternatively, too large elitism transfers rather ineffective decision strategies to the future generations that can slow the convergence to the needed solutions. If one is speaking about mutation rates, the following is valid in general. As all the mutations are highly random, their effect on searched optima can become rather unambiguous. However, mutation enriches the elite population with the new information, but can also add too much noise and so to prevent effective stabilization of the convergent results. Fig. 1 and Fig. 2 depict the situation that outlines the fixation of the parameters for the following analysis as follows: mutation rate 0.05, population 300, elitism 0.7.

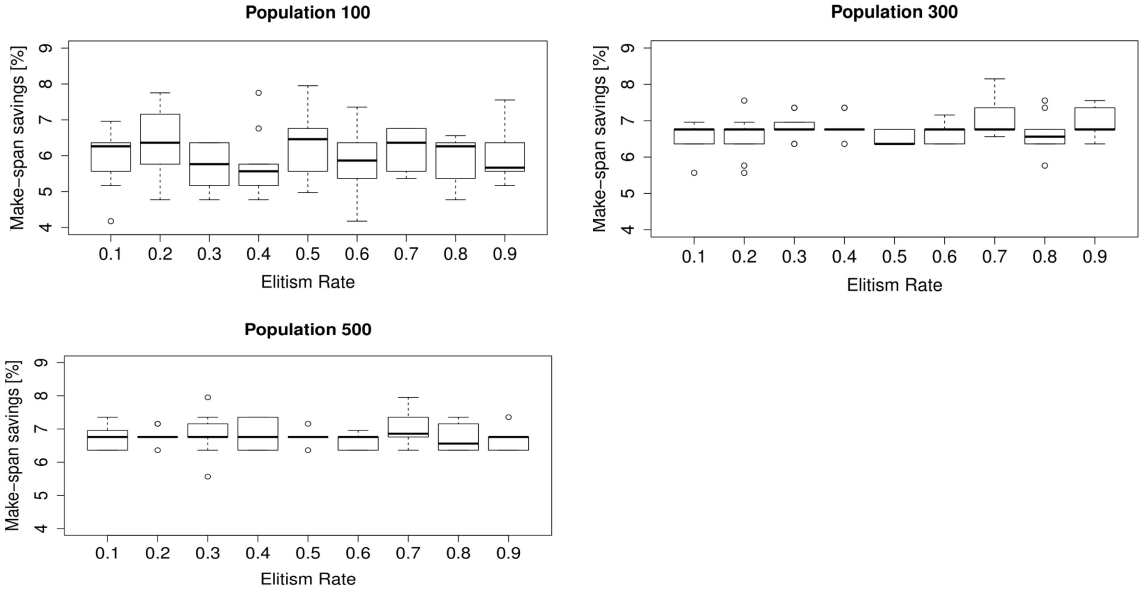

Fig. 1 Makespans distribution by different populations and elitism rates – 10 simulations (mutation rate 0.05; time savings against the Shortest processing time strategy)

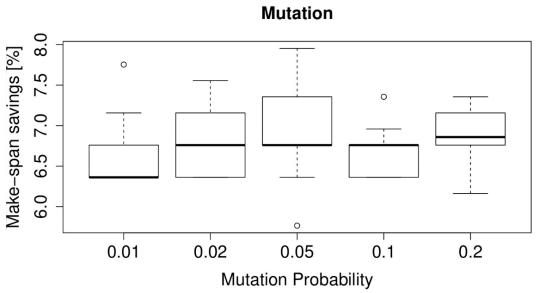

Fig. 2 Makespans distribution – 10 repeating simulations run by different mutation rates (elitism rate = 0.7; time savings against the Shortest processing time strategy)

4.2 Sensitivity to initial conditions, dynamics, and evolutionary stability

Any application of the GA deals with the stochasticity regarding (i) the initial population random settings; and (ii) random character of the mutations and crossing. It may lead to inconsistent computational results in repeating simulations, which makes performing of the computational stability analysis quite inevitable. In addition, the following research includes a simple investigation of the evolutionary dynamics to identify typical patterns in the evolutionary trajectory.

Following the principle of the previous analysis based on the repeating simulations using the same set of jobs and machines, 10 repeating simulation runs provide the results depicted in Fig. 3.

The genetic algorithm improves the autonomous decision mechanism of the self-organized machines in the first two simulation rounds rather significantly. One can distinguish two kinds of the simulation evolution. Smooth evolution process starts in the third iteration round lasting until the sixth one. This phase is typical with a quite small improvement of the makespan shortening. In the seventh round, the phase transition happened in the majority of the simulations. The makespan savings suddenly jumped by about 2 percent on the average. Then, the smooth evolution starts again and follows until the ninth round. After that, the phase transition happened again and the renewed machine decision rules demonstrate relevant improvements as the makespans significantly shortened on the average. The simulation round 15 starts the smooth evolutionary period, again, lasting with insignificant improvements until the end of the simulation, i.e. the 50th simulation round. The system reached its evolutionary equilibrium.

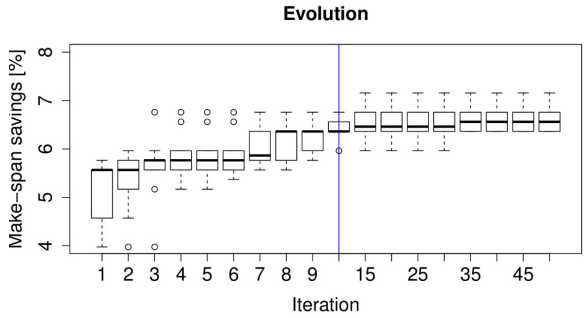

Fig. 3 Time evolution of the makespans savings – Boxplots of ten repeating simulations (time savings against the Shortest processing time strategy)

4.3 Changing list of the jobs – Robustness analysis

The above introduced analysis depicts the results of repeating simulations using the same list of jobs. On the other hand, one can object to the approach universality and its robustness against changing jobs list composition. To reduce the doubts, five random job list composition was generated (10 machines, 30 jobs, changing task requirements). The simulation parameters remain the same, i.e., elitism rate 0.8; mutation rate 0.05; no of iterations 50, population size 300.

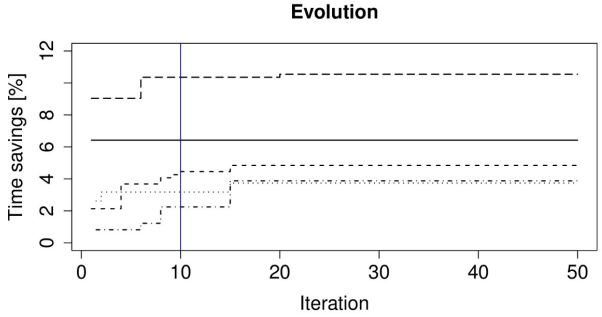

Fig. 4 The best makespans savings in case of 5 independent simulations with changing job list compositions (time savings against the Shortest processing time strategy)

Fig. 4 introduces the evolution of the best makespans within each of the five simulations. If compared the evolution led to the typical patterns given in previous analysis (see previous subsection and Fig. 3), evolution in the framework of the previously identified evolutionary pattern is obvious. The most significant changes happened in the first 10 simulation rounds alternating periods of the phase transition and quite stable development. The significant improvements of the best makespans were concentrated into the first 15 simulation rounds. On the other hand, there is the case of one simulation run, which did not demonstrate any improvement of the best decision during the whole simulation. It is possible if the randomly composed initial population contains at least one rather good production line chromosome that dominates during the whole evolutionary process. On the other hand, we consider this situation as considerably rare.

5. Conclusion

The technological complexity, shortening of the production cycles, and increasing specialization place high claims on the production planning and control of the production lines. At the same time, increasing of the IKT capacities and introduction of the artificial intelligence methods create new space for the yet unexperienced opportunities.

The presented research can be interpreted, of course, in the sense of a direction into a machine-to-machine economy (M2M economy). M2M economy means that machines are part of a system of interconnected machines without requiring human-to-human or human-to-machine interaction. At the same time, the machines are considered as autonomous participants. The simulation results prove that autonomous machines grouped in the production line can react flexibly to conditions arising in a given situation. In our case, every machine evaluates its situation according to its neighbours (predecessors and successors) and is independent in this decision-making process.

The presented research proposes a new form of the production line scheduling utilizing the Genetic algorithm methods in the framework of the machine line constituted from the independently deciding machines. Each machine makes its production decision based on the available somewhat restricted information; but flexible reacting to the capacity utilization of its neighbours in the technological process.

The presented research includes the details of the model proposal, model calibration, and analysis of the parameter settings. Sensitivity to the initial conditions is also included.

Despite using a quite simplifying example case, the proposed approach demonstrates a significant shortening of the makespans if compared to the Shortest processing time benchmark.

Acknowledgement

The paper was supported by the KEGA 002TUKE-4/2020, APVV-15-0358 and APVV-18-0368 projects.

References

[1] Maccarthy, B.L., Liu, J. (1993). Addressing the gap in scheduling research: A review of optimization and heuristic methods in production scheduling, *International Journal of Production Research,* Vol. 31, No. 1, 59-79, doi: 10.1080/00207549308956713.

[2] Çaliş, B., Bulkan, S. (2015). A research survey: Review of AI solution strategies of job shop scheduling problem, *Journal of Intelligent Manufacturing,* Vol. 26, No. 5, 961-973, doi: 10.1007/s10845-013-0837-8.

[3] Nouiri, M., Bekrar, A., Jemai, A., Niar, S., Ammari, A.C. (2018). An effective and distributed particle swarm optimization algorithm for flexible job-shop scheduling problem, *Journal of Intelligent Manufacturing,* Vol. 29, No. 3, 603-615, doi: 10.1007/s10845-015-1039-3.

[4] Hoogeveen, H. (2005). Multicriteria scheduling, *European Journal of Operational Research*, Vol. 167, No. 3, 592-623, doi: 10.1016/j.ejor.2004.07.011.

[5] Liu, J., Luo, X.-G., Zhang, X.-M., Zhang, F., Li, B.-N. (2013). Job scheduling model for cloud computing based on multi-objective genetic algorithm, *International Journal of Computer Science Issues*, Vol. 10, No. 1, 134-139.

[6] Zhang, G., Shao, X., Li, P., Gao, L. (2009). An effective hybrid particle swarm optimization algorithm for multi-objective flexible job-shop scheduling problem, *Computers & Industrial Engineering*, Vol. 56, No. 4, 1309-1318, doi: 10.1016/j.cie.2008.07.021.

[7] Sonmez, A.I., Baykasoglu, A. (1998). A new dynamic programming formulation of (n × m) flowshop sequencing problems with due dates, *International Journal of Production Research*, Vol. 36, No. 8, 2269-2283, doi: 10.1080/002075498192896.
[8] Garey, M.R., Johnson, D.S., Sethi, R. (1976). The complexity of flowshop and jobshop scheduling, *Mathematics of Operations Research,* Vol. 1, No. 2, 117-129, doi: 10.1287/moor.1.2.117.
[9] Błażewicz, J., Domschke, W., Pesch, E. (1996). The job shop scheduling problem: Conventional and new solution techniques, *European Journal of Operational Research*, Vol. 93, No. 1, 1-33, doi: 10.1016/0377-2217(95)00362-2.
[10] Ouelhadj, D., Petrovic, S. (2009). A survey of dynamic scheduling in manufacturing systems, *Journal of Scheduling*, Vol. 12, No. 4, 417-431, doi: 10.1007/s10951-008-0090-8.
[11] Mohapatra, P., Nayak, A., Kumar, S.K., Tiwari, M.K. (2015). Multi-objective process planning and scheduling using controlled elitist non-dominated sorting genetic algorithm, *International Journal of Production Research,* Vol. 53, No. 6, 1712-1735, doi: 10.1080/00207543.2014.957872.
[12] Supsomboon, S., Vajasuvimon, A. (2016). Simulation model for job shop production process improvement in machine parts manufacturing, *International Journal of Simulation Modelling,* Vol. 15, No. 4, 611-622, doi: 10.2507/IJSIMM15(4)3.352.
[13] Ma, D.Y., He, C.H., Wang, S.Q., Han, X.M., Shi, X.H. (2018). Solving fuzzy flexible job shop scheduling problem based on fuzzy satisfaction rate and differential evolution, *Advances in Production Engineering & Management,* Vol. 13, No. 1, 44-56, doi: 10.14743/apem2018.1.272.
[14] Ojstersek, R., Lalic, D., Buchmeister, B. (2019). A new method for mathematical and simulation modelling interactivity: A case study in flexible job shop scheduling, *Advances in Production Engineering & Management,* Vol. 14, No. 4, 435-448, doi: 10.14743/apem2019.4.339.
[15] Shen, X.-N., Yao, X. (2015). Mathematical modeling and multi-objective evolutionary algorithms applied to dynamic flexible job shop scheduling problems, *Information Sciences,* Vol. 298, No. 20, 198-224, doi: 10.1016/j.ins.2014.11.036.
[16] Meolic, R., Brezočnik, Z. (2018). Flexible job shop scheduling using zero-suppressed binary decision diagrams, *Advances in Production Engineering & Management,* Vol. 13, No. 4, 373-388, doi: 10.14743/apem2018.4.297.
[17] Yu, M.R., Yang, B., Chen, Y. (2018). Dynamic integration of process planning and scheduling using a discrete particle swarm optimization algorithm, *Advances in Production Engineering & Management,* Vol. 13, No. 3, 279-296, doi: 10.14743/apem2018.3.290.
[18] Holland, J.H. (1973). Genetic algorithms and the optimal allocation of trials, *SIAM Journal on Computing*, Vol. 2, No. 2, 88-105, doi: 10.1137/0202009.
[19] Holland, J.H., Reitman, J.S. (1978). Cognitive systems based on adaptive algorithms, *Pattern-Directed Inference Systems,* Vol. 1, No. 1, 313-329, doi: 10.1016/B978-0-12-737550-2.50020-8.
[20] Bierwirth, C., Kopfer, H., Mattfeld, D.C., Rixen, I. (1995). Genetic algorithm based scheduling in a dynamic manufacturing environment, In: *Proceedings of the Second Conference on Evolutionary Computation,* Perth, Australia, 439-443, doi: 10.1109/ICEC.1995.489188.
[21] Janes, G., Perinic, M., Jurkovic, Z. (2017). An efficient genetic algorithm for job shop scheduling problems, *Tehnički Vjesnik – Technical Gazette,* Vol. 24, No. 4, 1243-1247, doi: 10.17559/TV-20150527133957.
[22] Chen, W., Hao, Y.F. (2018). Genetic algorithm-based design and simulation of manufacturing flow shop scheduling, *International Journal of Simulation Modelling,* Vol. 17, No. 4, 702-711, doi: 10.2507/IJSIMM17(4)CO17.
[23] Zhang, J., Ding, G., Zou, Y., Qin, S., Fu, J. (2019). Review of job shop scheduling research and its new perspectives under Industry 4.0, *Journal of Intelligent Manufacturing,* Vol. 30, 1809-1830, doi: 10.1007/s10845-017-1350-2.
[24] Oral, M., Malouin, J.-L. (1973). Evaluation of the shortest processing time scheduling rule with truncation process, *AIIE Transactions*, Vol. 5, No. 4, 357-365, doi: 10.1080/05695557308974923.
[25] Koza, J.R. (1992). *Genetic programming: On the programming of computers by means of natural selection*, MIT press, Cambridge, USA.