Journal of Industrial and Systems Engineering
Vol. 7, No. 1, pp. 129- 140
Autumn 2014

JISE

Robust scheduling for three-machine robotic cell using interval data

Saiedeh Gholami[1*], Faezeh Deymeh[1]

[1]*Department of Industrial Engineering, K. N. Toosi University of Technology, Tehran, Iran*

s_gholami@kntu.ac.ir, faezeh.daymeh@gmail.com

Abstract

In reality, due to the lack of adequate environmental information, uncertainty is a common practice. In order to provide good and acceptable solutions, development of systematic methods for solving problems of uncertainty is important. One of these methods is based on robust optimization. This type of planning is to find a solution that is not sensitive to parameter fluctuations. In this article, a new way is represented to solve a three-machine robotic cell problem. An intervallic processing time is concerned as the problem being discussed. Different scenarios are defined by using robust optimization; afterwards, applying min-max regret method, robust counterpart of original problem is specified. Since the problem is NP-hard, a metaheuristic is applied to solve it. Genetic Algorithm (GA) as a population-based metaheuristic is employed. Cycle time and program operating time are calculated for different number of parts. It is demonstrated that by increasing the part numbers, gap between the robust and original cycle time increases. It is observed that both the cycle time and algorithm operating time increase.

Keywords: Robust optimization, min-max regret, cycle time, genetic algorithm

1-Introduction

Due to the intense competition established in industry, producers should reduce costs and increase efficiency to survive in this market. Material handling systems (between production stages) make great benefits like, increasing efficiency and processing rate, decreasing labor cost, and environmental contamination. Robots are used in many modern industries to displace parts in a cell (robotic cell). Many studies have been conducted about robotic cell concentrated on cell design, robot sequencing, and part scheduling.

In one of the primary works about robotic cells, a heuristic method for optimizing the working cycle of an industrial robot with two arms was presented (Bedini et al. 1979). Mioman and Nof (1985) studied cells with multiple robots. Wilhelm classified the computational complexity of assembly cells (Wilhelm, 1987). In the first studies, simulation was used to calculate cycle time. Kondoleon (1979) used computerized modeling for the simulation of robot movements. Claybourne (1983) used simulation to analyze effects of robot sequence on throughput. Agnetis (2000) found the optimal schedule for no-wait cells with two or three machines. Then, a polynomial algorithm was used to find an optimal two unit cycle with identical parts or with two different part types (Che et al. 2003). Lenver et al. (1997) studied no-wait cells with multiple robots. In one of the first works about interval robotic cell, branch and bound search was used (Lei and Wang, 1994).

*Corresponding author
ISSN: 1735-8272,

Branch and bound, linear programming, and two-valued graphs were used to find the optimal one-unit cycle (Chen et al. 1998); then, these techniques were applied for multi-unit cycles (Che et al. 2002). Based on the pickup criterion, cells are divided into free pickup, no-wait, and interval. Lenver et al. (1997) developed an algorithm which produces a one-unit cycle in a no-wait cell.

Manufacturing systems generally run in environments of high uncertainty with production schedules that are not fulfilled which are attributed to random interruptions, e.g. curtailed or imprecise job data, machine breakdowns, job cancellations, and rush orders. Although uncertainty is a highly realistic issue in developing jobs in shop floors, the majority of prior existing investigations employ deterministic models, which do not take uncertain deviations into consideration. The ignorance of uncertainty is regarded as a key source of the gap between theories and practice of classical scheduling. Consequently, some earlier studies suggested reactive or proactive methods to manage uncertainty in job scheduling (Sabuncuoglu and Goren 2009). The original job progression is adjusted or a new job sequence is generated on time when uncertainty occurs by using reactive techniques, whereas a robust initial schedule is generated before uncertainty occurrence by proactive approaches. As a result, proactive methods aiming to find results defiant to deviations, in contrary to those that passively react to uncertain disruptions, are more tending. Actually, robust scheduling aims to find a schedule that minimizes the effects of uncertainty on the objective function.

In dealing with robust approach a scenario set that contains all possible realizations of the parameters is defined. Numerous robust criteria can be employed to pick a solution (Roy, 2010). The most straightforward one is the min–max criterion, in which we select a solution minimizing the largest cost over all scenarios. A less conservative criterion is the min–max regret, in which a solution which minimizes the largest deviation from optimum over all scenarios is selected. Both criteria have been widely used in decision making under uncertainty (Luce and Raiffa, 1957). The scenario set can be specified in several ways. In this article the number of scenarios is a part of the input.

In this paper, a robotic cell in uncertain condition is considered. Three-machine robotic cell problem is formulated; afterwards, a genetic algorithm is used to calculate cycle time when processing time has a uniform distribution in [l, u] interval, this problem is also solved using worst case technique. Finally, the cycle time and algorithm running time for these two methods for different number of parts are compared.

2-**Problem definition**

As in the classification method for common scheduling problems, three elements must be determined: machine environment (α), processing characteristics (β), and objective function (γ) (Aytug et al. 2005).

2-1-Notation

The following notations are used to describe a robotic cell:

M_1, M_2, M_3 : Machines in a Robotic cell

M_0 : Input buffer

M_4 : Output buffer

δ : Robot travel time between two adjacent machines, i and i+1

ϵ : Time required for loading/ unloading at each machine

$$x_{ij} = \begin{cases} 1 & if\ job\ j\ is\ processed\ on\ machine\ i \\ 0 & else \end{cases}$$

A_i: The *i-th* robot activity (which is defined as unloading machine i, transfers a part from machine *i* to machine i+1, and loading machine i+1)

T_{S_1} : Cycle time obtained via $S_{1,3}$ robot movement cycle

T_{S_2} : Cycle time obtained via $S_{2,3}$ robot movement cycle

T_{S_3} : Cycle time obtained via $S_{3,3}$ robot movement cycle

T_{S_4} : Cycle time obtained via $S_{4,3}$ robot movement cycle

T_{S_5} : Cycle time obtained via $S_{5,3}$ robot movement cycle

T_{S_6} : Cycle time obtained via $S_{6,3}$ robot movement cycle

Three machines in a robotic cell with a single gripper robot used to move parts in a cell are considered here. The arrangement of machines in the cell is linear. No buffer is considered for intermediate storage. In this cell, n parts with different processing times are considered. Robot travel time between two adjacent machines is constant and equal to δ. Time required for loading/ unloading a part at each stage is a constant value equal to ϵ.

It is assumed that for many reasons parts processing time on a machine is not predictable and each part has a uniform distribution between 1 and 10 ($U\sim[1,10]$). Robust optimization is used to optimize objective function (minimize objective function) when processing time is intervallic.

There are many robot movement cycles for a three-machine robotic cell which produces one part type. Among these cycles, for simplicity we considered only one-unit cycles.

For three-machine robotic cell, there are six single-unit cycles ($S_1, S_2, S_3, S_4, S_5, S_6$). These cycles and their processing times are defined below.

S_1 Cycle: The first possible movement cycle named as S_1, in which robot has the following movement ($A_0A_1A_2A_3$). In this cycle, the first part is placed on the first machine, and robot waits for its processing on the first machine; afterwards, it transfers the part to the second and third machine, respectively. The cycle time for this kind of robot movement is:

$$T_{S_1} = 2\alpha + a + b + c \tag{1}$$

Where a, b, and c are processing time on the first, second, and third machine, respectively. α is equal to$4\delta + 4\epsilon$.

S_2 Cycle: The second possible movement cycle is known as S_2, in which robot has the following movement ($A_0A_2A_1A_3$). In this cycle, the first part is placed on the second machine. Robot loads a part on the first machine; next, it waits for the second machine to finish processing the part and unloads the second machine, transfers the part to the third machine; then, the third machine is loaded. Afterwards, robot unloads the first machine, transfers the part to the second machine and loads the second machine. The cycle time for this kind of robot movement is:

$$T_{S_2} = \alpha + \max[\beta, b, \beta/2 + a, \beta/2 + c, (a + b + c)/2] \tag{2}$$

Where a, b, and c are processing time on the first, second, and third machine, respectively. α and β are defined as follows: α=$4\delta + 4\epsilon$, $\beta = 8\delta + 4\epsilon$.

S_3 Cycle: In S_3 cycle the robot movement can be explained as ($A_0A_1A_3A_2$). At the beginning of this cycle, there is a part on the third machine. The robot takes a part from input buffer, transfers it to the first machine, loads the machine, and waits for the first machine to finish processing. Afterwards, it unloads the first machine and takes the part to the second machine then goes to third machine. Next, the third machine is unloaded and the part is transferred to the output buffer, then the robot returns to the second machine. Later, the robot unloads the second machine and transfers the part to the third machine and loads it. The cycle time for this kind of robot movement is:

$$T_{S_3} = \alpha + \max[c, \alpha + a + 2\delta, \alpha/2 + a + b] \quad (3)$$

S_4 Cycle: In S_4 cycle, the robot movement can be explained as ($A_0A_3A_1A_2$). At the beginning of this cycle, there is a part on the third machine. The robot takes a part from input buffer, transfers it to the first machine, then robot goes to the third machine and unloads it. Then, the robot goes to the output buffer and places the part in the output buffer. The robot returns to the first machine, unloads it and transfers the part to the second machine and waits for its processing. Afterwards, the robot unloads the second machine and loads the third machine. The cycle time for this kind of robot movement is:

$$T_{S_4} = \alpha + \max[\beta + b, \alpha/2 + a + b, \alpha/2 + b + c] \quad (4)$$

S_5 Cycle: In S_5 cycle, the robot movement can be explained as ($A_0A_2A_3A_1$). As this cycle starts, there is a part on the second machine. The robot takes a part from input buffer, transfers it to the first machine and loads it. Then, the robot goes to the second machine and unloads it. The robot takes the part to the third machine and waits for it to finish processing. The robot unloads the third machine and transfers the part to the output buffer. Later, the robot returns to the first machine, unloads it and transfers the part to the second machine and loads it. The cycle time for this kind of robot movement is:

$$T_{S_5} = \alpha + \max[a, \alpha + c + 2\delta, \alpha/2 + b + c] \quad (5)$$

S_6 Cycle: In S_6 cycle, the robot movement can be described as ($A_0A_3A_2A_1$). Firstly, there are two parts on the second and third machines. The robot takes a part from input buffer and transfers it to the first machine then loads the machine. Next, the robot goes to the third machine and unloads it. The part is taken to the output buffer. Subsequently, the robot goes back to the second machine, unloads it and transfers the part to third machine. Afterwards, the robot returns to the first machine and unloads it; the robot goes to the second machine and loads it. The cycle time for this kind of robot movement is:

$$T_{S_6} = \alpha + \max\{\beta, a, b, c\} \quad (6)$$

2-2-Problem Formulation

According to the above mentioned definitions, the following mathematical formulation is presented:

$$\text{P}: \quad \min Z = C \quad (7)$$

$s.t$:

$$C = min\{T_1, T_2, T_3, T_4, T_5, T_6\} \quad (8)$$

$$a = \sum_{j=1}^{n} x_{1j} p_j \quad (9)$$

$$b = \sum_{j=1}^{n} x_{2j} p_j \quad (10)$$

$$c = \sum_{j=1}^{n} x_{3j} p_j \quad (11)$$

$$x_{1j} + x_{2j} + x_{3j} = 1 \qquad \forall j \quad (12)$$

$$T_{S_1} = 2\alpha + a + b + c \quad (13)$$

$$T_{S_2} = \alpha + \max[\beta, b, \beta/2 + a, \beta/2 + c, (a + b + c)/2 \quad (14)$$

$$T_{S_3} = \alpha + \max[c, \alpha + a + 2\delta, \alpha/2 + a + b] \quad (15)$$

$$T_{S_4} = \alpha + \max[\beta + b, \alpha/2 + a + b, \alpha/2 + b + c] \quad (16)$$

$$T_{S_5} = \alpha + \max[a, \alpha + c + 2\delta, \alpha/2 + b + c \quad (17)$$

$$T_{S_6} = \alpha + \max\{\beta, a, b, c\} \quad (18)$$

$$x_{ij} = \begin{cases} 1 & if\ job\ j\ is\ processed\ on\ machine\ i \\ 0 & else \end{cases} \quad (19)$$

The first equation is the objective function aiming to minimize the cycle time. The second constraint defines the cycle time. Constraints from 9 to 11 calculate processing time on machines 1 to 3, respectively. Constraint 12 means that each part is processed on only one machine. Constraints 13 to 18 calculate the cycle time according to different robot movements. Constraint 19 defines x_{ij}as a binary variable.

3-Formulation of the robust three-machine robotic cell

In real world scheduling, there are uncertainties that cannot be predicted. One of these uncertain parameters is processing time. Whenever the probability distribution is not known, one of the best methods to deal with uncertainty is robust optimization.

In this paper, it is considered that there is uncertainty in processing time. Uncertainty is described through a set of scenarios. Each scenario describes a unique set of processing time.

If p_j^{λ} shows the processing time for job j in λ scenario, then the vector $P^{\lambda} = \{p_j^{\lambda} : j = 1,2, \dots, n\}$ shows the corresponding processing time of λ scenario.

If Ω shows the set of robot movement sequences, we have$\Omega = \{\sigma(1), \sigma(2), \dots, \sigma(6)\}$, in which $\sigma(i) = s_i$. In this condition, $\varphi(\sigma, P^{\lambda})$shows the cycle time for σ cycle, while having λscenario.

The best robot movement sequence, which has P^{λ} for processing time vector, must satisfy the following equation:

$$z^l = \varphi(\sigma_{\lambda}^*, P^{\lambda}) = \min_{\sigma \in \Omega} \varphi(\sigma, P^{\lambda}) \quad (20)$$

Here, objective function is defined as the minimization of the maximum regret for a specific part sequence. Regret for a set of processing times and for a certain movement sequence is defined as the difference between the best cycle time and the existing cycle time. So the robust schedule is defined as:

$$\min_{\sigma\in\Omega} max\{\varphi(\sigma, P^{\lambda}) - \varphi(\sigma_{\lambda}^{*}, P^{\lambda})\} \quad (21)$$

Different scenarios are defined as follows. For the *j-th* job in job sequence and all the jobs processed beforehand, the processing time is considered equal to the lower bound; for job $j+1, \dots n$ the processing time is considered equal to the upper bound (Luce and Raiffa, 1957).
If $k = 1,2, \dots, n$ is the counter of scenario, the robust counterpart for the mentioned problem is as follows:

$$\mathrm{P}_c: \quad \min Z = maxC_2 \quad (22)$$

$s.t$:

$$C = min\{T_1, T_2, T_3, T_4, T_5, T_6\} \quad (23)$$

$$C_2 = T_i - C \qquad \forall\ i = 1, \dots, 6 \quad (24)$$

$$a = \sum_{j=1}^{k} x_{1j} l + \sum_{j=k+1}^{n} x_{1j} u \quad (25)$$

$$b = \sum_{j=1}^{k} x_{2j} l + \sum_{j=k+1}^{n} x_{2j} u \quad (26)$$

$$c = \sum_{j=1}^{k} x_{3j} l + \sum_{j=k+1}^{n} x_{3j} u \quad (27)$$

$$x_{1j} + x_{2j} + x_{3j} = 1 \qquad \forall j \quad (28)$$

$$T_{S_1} = 2\alpha + a + b + c \quad (29)$$

$$T_{S_2} = \alpha + \max[\beta, b, \beta/2 + a, \beta/2 + c, (a+b+c)/2 \quad (30)$$

$$T_{S_3} = \alpha + \max[c, \alpha + a + 2\delta, \alpha/2 + a + b] \quad (31)$$

$$T_{S_4} = \alpha + \max[\beta + b, \alpha/2 + a + b, \alpha/2 + b + c] \quad (32)$$

$$T_{S_5} = \alpha + \max[a, \alpha + c + 2\delta, \alpha/2 + b + c \quad (33)$$

$$T_{S_6} = \alpha + \max\{\beta, a, b, c\} \quad (34)$$

$$x_{ij} = \begin{cases} 1 & if\ job\ j\ is\ processed\ on\ machine\ i \\ 0 & else \end{cases} \quad (35)$$

The first equation specifies the objective function (min-max regret), and constraint 24 defines regret. Constraints 25 to 27 calculate the processing time on machines according to different scenarios. All other constraints are like those for problem P.

4-Genetic algorithm

Genetic algorithm is used to solve the original problem and its robust counterpart.

4-1-Solution representation

The first step in applying a genetic algorithm is to represent a solution as a chromosome. In this article each chromosome consists of n genes. Each gene can get one of the values of 1, 2 or 3. If *j-th* job is performed on *i-th* machine, the *j-th* gene gets the value of i. An example of solution coding is showed in figure 1.

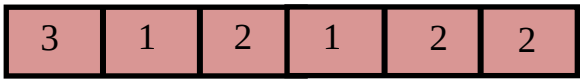

Fig 1. Solution representation

4-2-Initial population

After determining how to convert a solution to a chromosome, an initial population must be generated. In this article, the initial population is generated randomly.

4-3-Fitness function

For a certain solution (chromosome), the fitness function calculates its objective function. For the original problem fitness function is defined as the minimization of the cycle time and for its robust counterpart, fitness function is determined as the minimization of the maximum regret.

4-4-Selection methods

The selection method used in this article is roulette wheel. In this method, a chromosome with a greater fitness function has a greater probability to be chosen.

The selection process for roulette wheel can be explained as follows:

a. The selection probability, corresponding to each chromosome, is calculated as follows:
$$P_z = (\frac{1}{F_z})/sum(\frac{1}{F_z}) \quad (36)$$
where, P_z relates to the selection probability of z chromosome.
b. Chromosomes are arranged according to the value of P_z and the cumulative value of P_z is calculated.
c. A random number between zero and one is generated. If the random number belongs to each interval of cumulativeP_z, the chromosome corresponding to that interval is selected.

4-5-Genetic algorithm operators

Here, some operators are used to produce new solutions.

Cross over: After the selection process, an initial population of chromosomes is produced. Cross over operator can be used to create new solutions. For this operator, two random chromosomes are chosen first; secondly a position is chosen randomly for the cross over. In the third step, the chromosome values are replaced according to the cross over point.

Mutation: Mutation operator is used not to intensify rapidly, and prevent getting captured at local optimums. In this article, to mutate a chromosome (solution) a gene is chosen randomly; then a random number among 1, 2 or 3 is chosen and its value is replaced with the initial value of gene.
The schematic of GA used in this article is shown in figure 2.

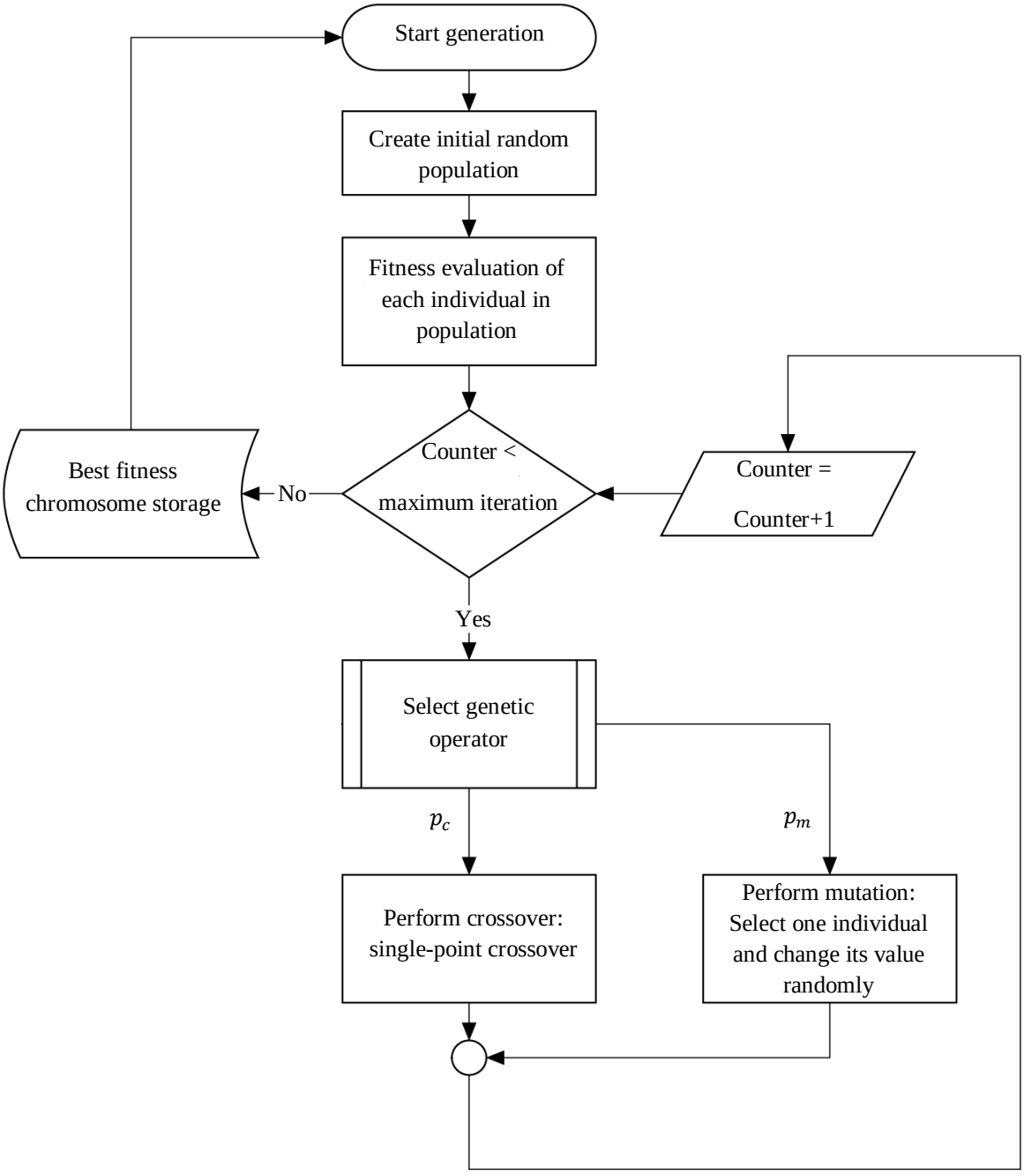

Fig 2. GA schematic

5-Results

To solve this problem it is considered that (robot loading/unloading time) and (robot travel time) are fixed and equal to 0.5 and 2, respectively. It is assumed that processing time is a random number between a lower bound and upper bound. The assumptions for the GA parameters are stated in table 1.

Table 1. GA parameters value

Population size	50
P cross over	0.8
P mutation	0.1
Number of jobs	5-50

The proposed solution methods have been examined in terms of both solution quality and CPU time. A laptop with an Intel Core i5 CPU and 4 Gb RAM was used to solve the problem. The algorithm was encoded using Matlab R2008a. The algorithm was implemented 100 times for each number of parts. The results are shown in table 2. For a better comparison, the results are depicted on figures 1 and 2.

Table 2. GA results (OFV=Objective Function Value)

Problem name	number of parts	best OFV	worst OFV	average OFV	average running time (seconds)
P	5	24	28	26.130927	4.392338
P_c (minmax regret)		35.5912	50	43.790275	12.422024
P	10	28	35.9046	29.47995	4.466567
P_c (minmax regret)		55.408	76.4319	65.937225	21.45274
P	15	31.6146	4.6572	38.161759	4.463692
P_c (minmax regret)		76.3041	103.7656	90.323025	31.37196
P	20	38.2328	4.5596	46.8046	4.349605
P_c (minmax regret)		87.6124	130.0544	112.24127	45.78947
P	25	47.3136	4.6117	56.498416	4.472772
P_c (minmax regret)		115.1484	158.4332	136.427879	74.135499
P	30	55.4134	77.7449	64.98958182	4.474119
P_c (minmax regret)		137.3227	107.8956	159.719371	92.254558
P	35	62.1198	89.1647	73.957285	4.487917
P_c (minmax regret)		157.1999	198.4047	182.294121	112.101008
P	40	69.4966	4.6041	83.189408	4.501832
P_c (minmax regret)		177.4728	229.3319	205.356919	95.86956
P	45	77.0744	4.6103	93.753539	4.535005
P_c (minmax regret)		205.9645	257.504	230.746583	112.362817
P	50	87.8246	117.632	102.200716	4.686501
P_c (minmax regret)		223.5369	159.5506	250.35459	139.392433

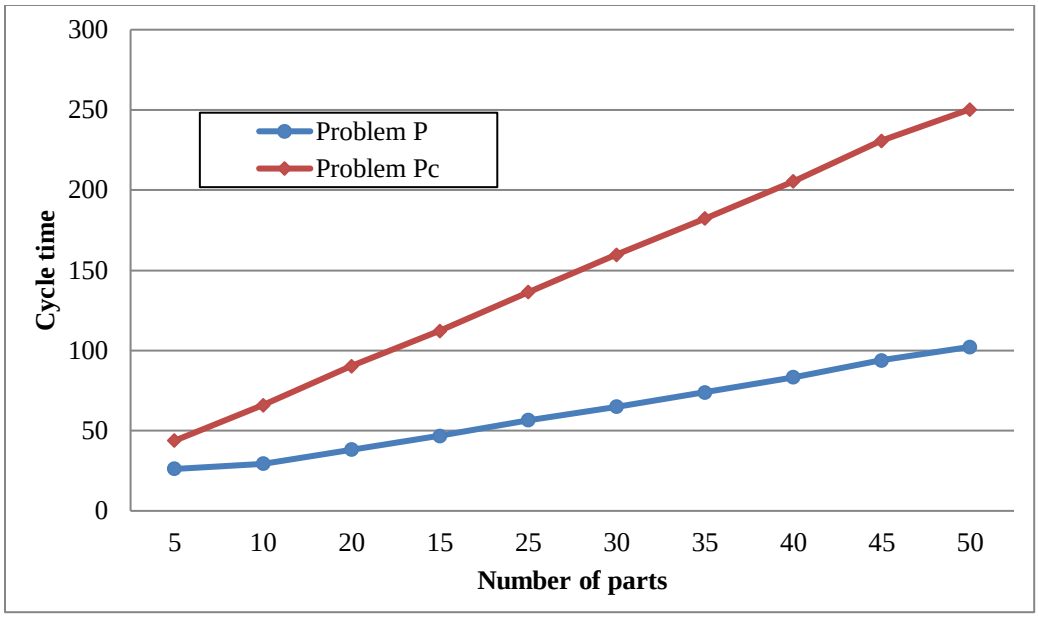

Fig 3. The gap of cycle time for the original and robust counterpart problem

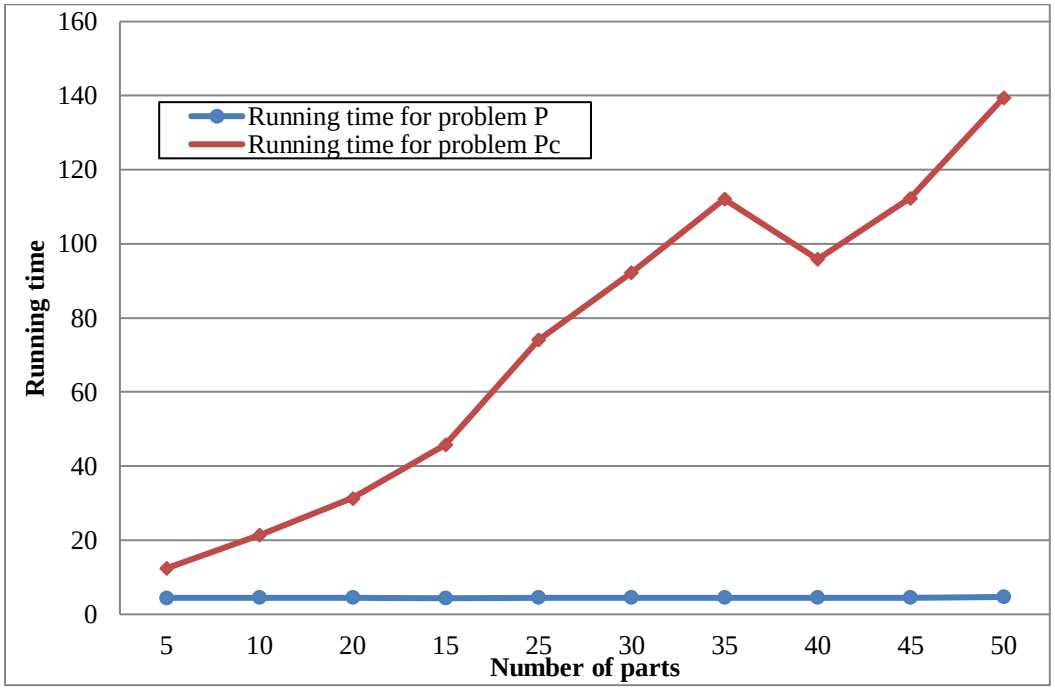

Fig 4. The gap of running time for the original and robust counterpart problem

6-Conclusion and discussion

In this article a new three-machine robotic cell problem has been considered. This problem deals with an intervallic processing time. Later, the original and the robust form of the problem have been formulated. The two problems are solved, using genetic algorithm. For different number of parts the cycle time and program passing time, have been calculated. Both cycle time and algorithm running time has increased. It is shown that by increasing the part numbers, the gap between the two methods increases. Since a robust solution worsens the performance measure, considering a multi-criteria objective function, which minimizes the cycle time and maximizes the benefit, can be a good choice for future studies.

References

Agnetis A. (2000), Scheduling No-Wait Robotic Cells with Two and Three Machines; *European Journal of Operational Research* 123;303–314.

Aytug H., Lawley M.A., McKay K., Mohan S., Uzsoy R. (2005), Executing production schedules in the face of uncertainties: A review and some future directions; *European Journal of Operational Research*161;86–110.

Bedini R., Lisini G.G., Sterpos P. (1979), Optimal Programming of Working Cycles for Industrial Robots; *Journal of Mechanical Design. Transactions of the ASME* 101; 250–257.

Che A., Chu C., Chu F. (2002), Multicyclic Hoist Scheduling with Constant Processing Times; *IEEE Transactions on Robotics and Automation*18; 69–80.

Che A., Chu C., Levner E.(2003), A Polynomial Algorithm for 2-degree Cyclic Robot Scheduling; *European Journal of Operational Research* 145;31–44.

Chen H., Chu C., Proth J. (1998), Cyclic Scheduling with Time Window Constraints; *IEEE Transactions on Robotics and Automation* 14;144–152.

Claybourne B.H. (1983), Scheduling Robots in Flexible Manufacturing Cells; *CME Automation* 30;36–40.

Kondoleon A.S. (1979), Cycle Time Analysis of Robot Assembly Systems; *Proceedings of the Ninth Symposium on Industrial Robots;*575–587.

Lei L., Wang T.J. (1994), Determining Optimal Cyclic Hoist Schedules in a Single- Hoist Electroplating Line; *IIE Transactions* 26;25–33.

Levner E., Kats V., Levit V. (1997), An Improved Algorithm for Cyclic Flowshop Scheduling in a Robotic Cell; *European Journal of Operational Research* 97;500–508.

Luce R.D., Raiffa H. (1957), Games and Decisions: Introduction and Critical Survey; Dover Publications Inc.

Maimon O.Z., Nof S.Y. (1985), Coordination of Robots Sharing Assembly Tasks; *Journal of Dynamic Systems Measurement and Control. Transactions of the ASME* 107; 299–307.

Roy B. (2010), Robustness in operational research and decision aiding: A multi-faceted issue; *European Journal of Operational Research*200; 629–638.

Sabuncuoglu I., Goren S. (2009), Hedging production schedules against uncertainty in manufacturing environment with a review of robustness and stability research. *International Journal of Computer Integrated Manufacturing*22; 138–57.

Wilhelm W.E. (1987), Complexity of Sequencing Tasks in Assembly Cells Attended by One or Two Robots; *Naval Research Logistics* 34;721–738.