Jurnal Teknik Industri, Vol. 22, No. 2, December 2020
ISSN 1411-2485 print / ISSN 2087-7439 online
DOI: 10.9744/jti.22.2.85-92

A Genetic Algorithm for the Double Row Layout Problem

Achmad Pratama Rifai[1]*, Setyo Tri Windras Mara[1], Putri Andriani Kusumastuti[1], Rakyan Galuh Wiraningrum[1]

Abstract: The double row layout problem (DRLP) is an *NP*-hard and has many applications in the industry. The isuue concerns arranging the position of $\boldsymbol{n}$ machines on the two rows so that the material handling cost is minimized. Although several mathematical programming models and local heuristics have been previously proposed, there is still a requirement to develop an approach that can solve the problem efficiently. Here, a genetic algorithm is proposed, which is aimed to solve the DRLP in a reasonable and applicable time. The performances of the proposed method, both its obtained objective values and computational time, are evaluated by comparing it with the existing mathematical programming model. The results demonstrate that the proposed GA can find relatively high-quality solutions in a much shorter time than the mathematical programming model, especially in the problem with a large number of machines.

Keywords: Facility layout planning, double row layout problem, genetic algorithm, material handling cost.

Introduction

Facility layout planning is a complex manufacturing environment that optimizes space usage and machine layout configuration. It is an essential task required to operate the production and service systems efficiently, since a well-designed facility layout could reduce many forms of waste, such as excessive motion, waiting time, and material handling. In this regard, it is observed by Mohamadghasemi and Vencheh [1] that 20-50% of total operating cost is contributed by the cost of material handling, which implies the significance of facility layout planning tasks.

Material handling cost is essential as it is the main cost involved in designing and operating a material handling system. The cost consists of freight, insurance, customs clearance, and so on. An effective space usage requires low material handling costs, and these costs may be reduced by determining product flows via optimizing and minimizing adjacent machines' clearance. In the machine layout, there should be a clearance or minimum separation distance required between the machines. The clearance may vary between the machines, depending on the sequence. However, in the practical problems, the clearance is subject to the machine, such as the length of the machine and function. Therefore, it is crucial to put a similar machine close to each other for the maintenance.

[1] Faculty of Engineering, Mechanical and Industrial Engineering Department, Universitas Gadjah Mada, Jl. Grafika 2, Yogyakarta 55281, Indonesia. Email achmad.p.rifai@ugm.ac.id, setyotriw@ugm.ac.id, putriadriani99@mail.ugm.ac.id, ragyangaluh@mail.ugm.ac.id

* Corresponding author

If the clearances between machines are not equal to each distance, it will affect the clearances values and difficulty finding the exact location.

This DRLP is the problem of allocating a given set of machines on both sides of a straight-line corridor to minimize the total cost of transporting materials among machines. It is different from Single Row Facility Layout Problem (SRFLP) which concerns the facilities' arrangement on a single row (Amaral [2]) where all the machines are placed at the same side of the corridor. The DRLP aims to determine the machine locations in the two rows. The objective function is the total flow cost while ensuring that there will be no overlaps of machines. The objective is subject to some constraints, such as the width of the machines and the number of products processed by each machine.

Several approaches have been developed to solve the DRLP. Amaral [3] formulated a mixed-integer programming (MIP) where the constraint sets are a non-overlapping set of machine and distance measuring constraints set between pairs of machines in the row. For measuring the constraints set in different rows, MIP assigns the variables to specify the machine locations whether it will be placed on the upper or lower row. The proposed MIP was then compared with previous models by Chung and Tanchoco [4] and Amaral [5]. Further, Secchin and Amaral [6] updated the MIP model for more efficient computation.

As indicated by previous research, the use of mathematical programming is limited as it is not suitable for large datasets. Besides, it generally requires long computation time even for small and medium sizes of instances. Meanwhile, the previously

developed heuristic methods were mainly based on the combination of local heuristics with linear programming (Murray *et al.* [7]; Zuo *et al.* [8]; Wang [9]; Amaral [10]). Although these methods can reduce the computation time significantly, the involvement of LP limits its efficiency. Due to this issue, the use of a metaheuristic is expected to provide reasonably good solutions in short computation time. Therefore, we propose the use of a genetic algorithm (GA) to produce solutions and find the near-optimal solution. The GA approach aims to optimize the machine arrangement with shorter computation times, which then enables the solving of the DRLP with a large number of machines. The studies on the DLRP is still on infancy, and to the best of our knowledge, most studies applied the combination of heuristics and mathematical programming. Therefore, this study is the first to consider a Genetic Algorithm to solve the DRLP.

Methods

Problem Formulation

This section explains the model for DRLP as outlined by Amaral [10]. Several assumptions holds true in this model: (i) The corridor width is negligible and only distance along the x-axis is considered, (ii) The loading/unloading point is at the center of each machine, (iii) The clearance are included in the length of the machines, and (iv) The product flows follow symmetric matrices. The simple illustration of double row layout with 7 machines is presented in Figure 1.

There are n machines to be arranged to form the double row layout. Each machine has a distinct length $l_i, i \in N$ where $N = \{1, \dots, n\}$ is the set of machines. As mentioned in the assumption (iii), the length of machines includes the required minimum clearance between two machines. Thus, the actual length of the machines is lower than the listed length for the computation. There are several materials to be processed by the machines in which each product may require a different subset of machines to process. Subsequently, the materials must be transported between one machine to another required machine until finishing all processes.

Borrowing the description of Amaral [3], the mathematical formulation of DRLP can be presented as a mixed-integer linear programming model. There are two binary decision variables required. t_{ij} is a binary variable which takes the value of 1 if machine j is located to the right of machine i, and 0 otherwise. This decision variable is employed to avoid any overlap between machines located in the same row.

Figure 1. Feasible double row layout with seven machines

Then, u_{ij} is a binary variable to indicate the row of a given machine, it takes the value of 1 if machine i and machine j are placed at the same row, and 0 otherwise. Finally, using the sets, notations, and parameters from Table 1, the formulation can be drawn as below. The objective function is minimizing the material handling cost f. To simplify the problem, only the operating cost of material handling is considered. Meanwhile, the cost of product damage and maintenance is not examined since the material handling system mainly causes it, thus not relevant for the layout planning problem. Besides, the equipment and unit purchase costs are omitted as it is assumed that there is no change in material handling equipment.

Table 1. Description of sets, notations, and parameters

Symbol	Description
n	number of available machines
N	set of machines, $\{1, \dots, n\}$
l_i	length of machine
L	sum of the length of all machines, $L = \sum_{i \in N} l_i$
x_i	abscissa or horizontal location of the center of machine i
c_{ij}	material flow between machines i and j
d_{ij}	distance between the centers of machine i and machine j

Objective function:

$$\min f = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} c_{ij} d_{ij} \quad (1)$$

Subject to:

$$d_{ij} \geq x_i - x_j \quad (2)$$

$$d_{ij} \geq x_j - x_i \quad (3)$$

$$x_j + \left(\frac{l_i + l_j}{2}\right) \leq x_i + L\left(1 + t_{ij} - u_{ij}\right) \quad (4)$$

$$x_i + \left(\frac{l_i + l_j}{2}\right) \leq x_j + L\left(2 - t_{ij} - u_{ij}\right) \quad (5)$$

$$d_{ij} \geq \left(\frac{l_i + l_j}{2}\right) u_{ij} \quad (6)$$

The equations (2)-(6) are applied for every $i \in N \backslash \{j\}, j \in N \backslash \{i\}, i < j$

$$d_{ij} - d_{jk} - d_{ik} \leq 0 \quad (7)$$

$$-d_{ij} + d_{jk} - d_{ik} \leq 0 \quad (8)$$

$$-d_{ij} - d_{jk} + d_{ik} \leq 0 \quad (9)$$

$$u_{ij} + u_{jk} + u_{ik} \geq 1 \quad (10)$$

$$-u_{ij} + u_{jk} + u_{ik} \leq 1 \quad (11)$$

$$u_{ij} - u_{jk} + u_{ik} \leq 1 \quad (12)$$

$$u_{ij} + u_{jk} - u_{ik} \leq 1 \quad (13)$$

The equations (7)-(13) are applied for every $i \in N\backslash\{j,k\}, j \in N\{i,k\}, k \in N\backslash\{i,j\}, i < j < k$

$$\frac{l_i}{2} \leq x_i \leq L - \frac{l_i}{2};\ \forall\, i \in N \tag{14}$$

$$x_{i^*} \leq x_{j^*} \tag{15}$$

$$i^* = \underset{i \in N\backslash\{j\}}{\arg max}\{\sum_{j \in N\backslash\{i\}} c_{ij}\} \tag{16}$$

$$j^* = \underset{j \in N\backslash\{i^*\}}{\arg min}\{c_{i^*,j}\} \tag{17}$$

$$t_{ij}, u_{ij} \in [0,1]; \forall\, i \in N\backslash\{j\}, j \in N\backslash\{i\}, i < j \tag{18}$$

The objective function (1) is to minimize the material handling cost f among all machines in the layout, which is calculated by the product of the material flow and the distance between machines. Here, the distance is calculated between the centers of the machines. This objective is subjected to a set of constraints. Equations (2) and (3) calculate the distance between machines. Equations (4) and (5) prevent the overlap between machines if these machines are located at the same row. Equations (6) gives the lower bound value for the distance variables d_{ij}. Equations (7), (8), and (9) are the valid inequalities on distance variables d_{ij}. Similarly, Equations (10), (11), (12), and (13) are the valid inequalities on the variables to control the positional row of a given machine. Equations (14) provides the bounds to the value of abscissa location. Equation (15) is the symmetry-breaking constraint. Within each case of the DRLP, there are two symmetric optimal solutions since it does not matter at which machine the process is initially started. Thus, this equation is aimed to eliminate one of the symmetric solutions by stating that $x_{i^*} \leq x_{j^*}$, where i^* corresponds to a machine with the largest total flow to the other machines and j^* corresponds to a machine with the least flow to the i^*. The calculations of i^* and j^* are presented in Equation (16) and Equation (17). Last, Equation (18) limits the decision variables t_{ij} and u_{ij} to a binary value.

Genetic Algorithm Design

This section describes the detail of genetic algorithm (GA) designed to solve the DRLP problem. The GA is a stochastic search technique that mimics the mechanisms of Darwinian evolution based on the concept of the survival of the fittest (Deb [11]; Goldberg [12]). The previous application of GA indicates its favorable performance in manufacturing system optimization, such as for FMS scheduling problem in loop layout (Rifai *et al.* [13]), layout planning in SRLP (Datta *et al.* [14]), and unequal area production facilities planning (Hou *et al.* [15]). GA uses a selection operator that is created by a random number and selection probability calculated by the objective values of chromosomes in the population, and a mating pool is created with the selected chromosomes from the original population.

	M_1	M_2	M_3	M_4	M_5	M_6	M_7
Row	1	2	1	1	2	2	1
Machine Index	3	1	1	4	3	2	2

Figure 2. Solution representation

Crossovers are induced by random numbers for pairs of chromosomes in the pool, and then chromosomes are repaired if necessary. A mutation creates an exchange of locations in a chromosome and it is also triggered based on a random number.

It begins with a set of random individuals, referred to as a population, which is evolved over iterations by some repeated applications of some genetic operators, such as selection, crossover, and mutation. The most crucial component of a GA is the solution representtation, popularly known as the chromosome or individual which represents a complete solution of a problem. Here, we use two-rows representation. The first row represents the layout row and the second presents the machine index. Figure. 1 presents an example of solution representation for $n = 7$.

Figure. 2 presents an example of solution representation for $n = 7$. Thus, for example, machine 1 is placed on the first row, and positioned at the 3rd sequence from the starting point after machine 3 and machine 7. This type of representation allows the GA operators to modify either the row of machine or the position of the machine inside the row. Therefore, it can balance the exploration and exploitation of the search space.

The proposed GA creates an initial population randomly and evaluates the chromosome, then the population is sorted based on the objective values. We decide the constraints as material flow costs, the width of the machines, and processed material per machine. The objective function is the total flow cost while ensuring that there will be no overlaps of machines. The concept of the selection is emphasizing good individuals and eliminates the weak ones by forming a temporary population, known as the mating pool. The mating pool consists of the selected chromosomes from the original population. Then, crossover is performed on the selected parents and generated offspring are evaluated.

In this case, we adopt the one-point crossover. One-point crossover works by selecting single point fragmentation in the parent chromosomes and combines the parent genes to create an offspring. The two parents are selected from the population by using roulette wheel selection. Two offspring are created by combining the parent genes at a crossover point. In this study, the one-point crossover requires two phases in order to create the offspring.

				Solution 1	Solution 2
(a)	Row exchange only	Parents	Row	1 2 1 1 2 2 1	2 2 1 1 1 1 2
			Machine index	3 1 1 4 3 2 2	1 2 3 4 2 1 2
		Offspring	Row	1 2 1 1 1 1 2	2 2 1 1 2 2 1
			Machine index	1 1 5 4 3 2 2	3 4 3 1 2 1 2
(b)	Machine index exchange only	Parents	Row	1 2 1 1 2 2 1	2 2 1 1 1 1 2
			Machine index	3 1 1 4 3 2 2	1 2 3 4 2 1 2
		Offspring	Row	1 2 1 1 2 2 1	2 2 1 1 1 1 2
			Machine index	3 3 1 4 2 1 2	1 3 2 4 3 2 2
(c)	Row and machine index exchange	Parents	Row	1 2 1 1 2 2 1	2 2 1 1 1 1 2
			Machine index	3 1 1 4 3 2 2	1 2 3 4 2 1 2
		Offspring	Row	1 2 1 1 2 2 1	2 2 1 1 1 1 2
			Machine index	3 3 1 4 2 1 2	1 3 1 4 3 2 2

Figure 3. Procedures of one-point crossover

		M_1	M_2	M_3	M_4	M_5	M_6	M_7
Original chromosomes	Row	1	2	1	1	2	2	1
	Machine Index	3	1	1	4	3	2	2
(a) Row exchange only	Row	1	2	2	1	2	1	1
	Machine Index	3	2	1	4	3	2	3
(b) Machine index exchange only	Row	1	2	1	1	2	2	1
	Machine Index	3	2	2	4	3	1	1
(c) Row and machine index exchange	Row	1	2	2	1	2	1	1
	Machine Index	3	1	2	4	3	1	2

Figure 4. Procedures of swap mutation

First, the type of solution modification is determined, either exchanging the genes for machine index, row, or both of them. Second, the crossover point is selected randomly using uniform distribution $CP = U(1, n-1)$. Afterward, the genes of the two parents after the crossover point are exchanged to form new offspring. Figure. 3 presents the procedures for one-point crossover for this study.

After creating the new offspring, the proposed GA will select the newly generated solutions in order to apply the mutation method. Mutation is used to find the global optimum derived from the randomly chosen children with another predefined probability, known as mutation probability. It applies low mutation probability and then swaps the values of the two elements of the child individual. Swap mutation selects the two parents randomly and swaps their positions. The genes can be the same or different routes. For instance, consider the following chromosome in Figure. 2. If positions 3 and 6 are selected, then the machine 3 and machine 6 would interchange their gene values. Similar to the crossover operators, the exchanging values may involve only the row, the machine index, or both information. Then the mutated chromosome will be depicted in Figure 4.

The need for mutation comes from the fact that the loss of successive generations could discard good genetic material forever. By performing occasional random changes in the chromosomes, GA ensures that new parts of the search space are reached, which selection and crossover alone could not fully guarantee. By doing so, swap mutation ensures that no important features are prematurely lost, thus maintaining the mating pool diversity. Since the solutions represented using the two-rows representation, the layout row and the machine index, the crossover and mutation operators are tailored to accommodate this representation. The mechanism for both operators is developed by considering the need to balance between the exploration and exploitation of our proposed GA algorithm in searching for a global optimum solution.

The crossover and mutation operators can modify the solution either by transferring the selected machines into other rows for exploration of the search space or changing the machine index only (the machine is still positioned in the same row) for exploitation of promising neighborhood. After the crossover and mutation, the newly generated solutions are adjusted to ensure no constraints are violated. The adjustment is performed by prohibiting the machines in the same row to have the same index. In such case, one of the redundant indices is simply changed to the new index. After the solution adjustment process, the best solution among chromosomes is recorded. The procedure of the proposed GA is depicted in the Algorithm 1.

Algorithm 1: Genetic Algorithm for DRLP

Input: an instance (n, l, d), GA parameter (P, T, cr, mr)
Initialize feasible solutions $x_i, i = \{1, 2, \dots, P\}$
Insert x_i to A
Evaluate the fitness f_i of each chromosome x_i
Record the best solution x_b, where $f_b = f_i$
set $s = 1$
While $s \leq T$ **do:**
    Perform roulette wheel to select parents $p_i, i \in A$
    for $j = 1\ to\ P/2$ **do:**
        If $U(0,1) < cr$ **then:**
            Determine cutting point $CP = U(1, n-1)$
            Perform one-point crossover
    for $k = 1\ to\ P$ **do:**
        If $U(0,1) < mr$ **then:**
            Determine the swap position
            Perform swap mutation
    Perform solution adjustment
    Insert $x_i{}'$ to A'
    Evaluate the fitness $f_i{}'$ of each offspring $x_i{}'$
    Record the best solution $x_b{}'$, where $f_b{}' = f_i{}'$
    if $f_b{}' < f_b$
        $x_b = x_b{}'$
        $f_b = f_b{}'$
    Replace the population $A \leftarrow A'$
    Update $s = s + 1$
Output: best solution x_b, minimum objective value f_b

Results and Discussions

The computation experiments are executed on an Intel® Core™ i7-10770 CPU 2.90 GHz with 32 GB of RAM with a Windows 10 operating system, and the proposed GA is developed on Python 3.7.6. To assess the performance of the GA, comparisons with the results of MIP are performed on several datasets. The instances for benchmarking are n =9,10,11 (Simmons [16]); $n = 11,12,13$ (Amaral [4]); $n = 14$ (Secchin and Amaral [6]); $n = 15$ (Amaral [17]); $n = 17$ (Amaral [2]); and $n = 18$. The complexity of the datasets is determined by the number of machines n, where the increase on the number of machine means more possible solutions thus enlarge the search space. The parameter settings of the proposed GA are population size $P = 100$, maximum iteration $T = 3000$, crossover probability $cr = 0.95$, and mutation probability $mr = 0.1$. Table 2 presents the results of computation experiments on all instances.

The optimal values and the computation time of MIP are obtained from Amaral [5]; Amaral [3]; and Secchin and Amaral [6]. The differences Δ are calculated as the percentage deviation between the optimal value and result of GA, indicated as follow.

$$\Delta = \frac{f_{GA} - f_{optimal}}{f_{optimal}} \times 100\% \quad (19)$$

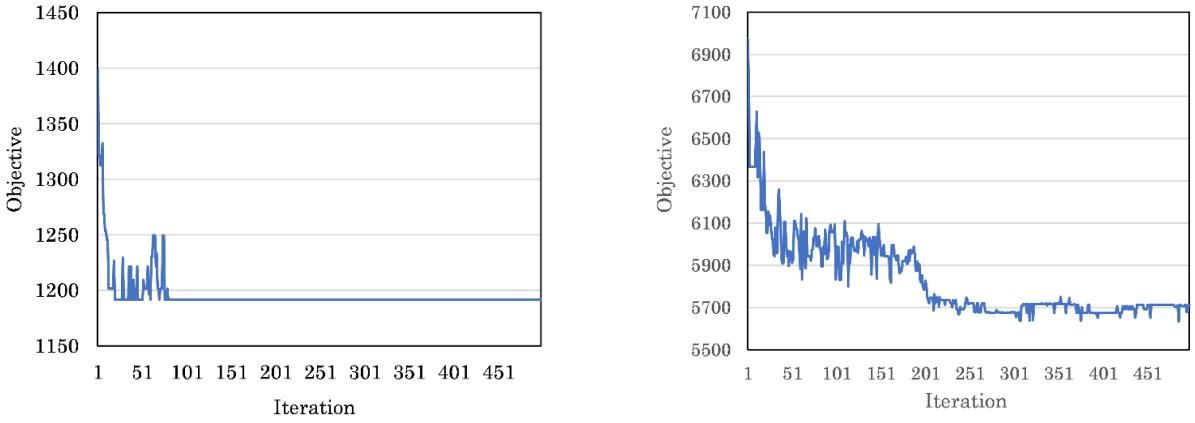
where f_{GA} is the objective value obtained by GA, while $f_{optimal}$ is the optimal objective value. The results indicates that the proposed GA return similar objective values as compared to the optimal values in all instances with average differences Δ is less than 1.68%. This indicates that the proposed operators and solution adjustment process successfully explores the neighborhood of an individual by preserving its feasibility. In this way, the population is gradually improved towards the optimum value of a given objective function. Further, Figure. 5 shows the best obtained fitness of each generation in instance S9 and P18.

The graph is cut until iteration 500 since there is no significant improvement beyond this iteration. To avoid premature convergence, we set mild mutation probability. Hence, although the algorithm has started to reach convergence, the mutation ensures that there is enough variability of solutions in each generation. The results show that the obtained objectives by GA are very close to the optimal value, which indicates that it can obtain the near optimal solution. The slight differences are actually caused by the assumption that both rows start at the same point. To minimize the traveled distance, the starting point of the shorter row can further be adjusted.

Figure 6 depicts the comparison on computational time. The strength of the proposed GA lies in its computational speed as it can deliver good solutions in much shorter time than the MIP model. The out 27 instances (S9, S10 and am11f), which are among the simplest problems. The differences on the computation time increases as the instance complexity increases. The computation time of GA is relatively stable across various complexity as indicated in Fig. 5, ranging from 37.2s for the simplest instances to 75.70s for the most complex instances. It is found that the bigger portion of computational time for the GA is spent for fitness evaluation with average of 49.3% of the computation time is for solution adjustment, calculation of the distance between machines, and calculation of the objective function. The crossover and mutation operators consume 23.3% and 0.6% of computation time respectively, while the rest is for other processes such as for initial population generation, parent selection, and preserving the best solution. Hence, since the number of solutions generated by GA are uniform in all instances, the computational times of GA do not differ significantly. On the other side, the increase on complexity significantly elevate the computational time of MIP which causes MIP is unable to solve more complex instances, i.e. P17 and P18, in reasonable time. Meanwhile, the proposed GA can solve these instances efficiently, thus confirming its suitability to be applied for the DRLP with larger number of machines. computation time of MIP is faster than GA only in 3.

Table 2. Results of computation

No	Instance	n	Objective values			Time (s)	
			Optimal	GA	Δ	MIP	GA
1	S9	9	1179	1181.5	0%	7.96	37.72
2	S9H	9	2293	2294.5	0%	104.73	37.30
3	S10	10	1351	1378.5	2%	32.7	40.83
4	S11	11	3424.5	3513.5	3%	325.36	44.65
5	am11a	11	5559	5696.5	2%	343.9	45.76
6	am11b	11	3655.5	3683.5	1%	277.52	46.11
7	am11c	11	3832.5	3858.5	1%	271	44.65
8	am11d	11	906.5	926.5	2%	69.26	45.36
9	am11e	11	578	584	1%	63.51	44.75
10	am11f	11	825.5	855	4%	31.64	44.94
11	am12a	12	1493	1529	2%	374.63	48.46
12	am12b	12	1606.5	1657.5	3%	403.92	48.58
13	am12c	12	2012.5	2072	3%	435	49.12
14	am12d	12	1107	1117	1%	115.37	48.53
15	am12e	12	1066	1090	2%	120.45	48.82
16	am12f	12	997.5	1032	3%	150.35	48.77
17	am13a	13	2456.5	2478.5	1%	12,481.17	52.46
18	am13b	13	2864	2891	1%	10,935.62	52.93
19	am13c	13	4136	4257	3%	1,697.90	52.81
20	am13d	13	6164.5	6274.5	2%	3,380.06	52.93
21	am13e	13	6502.5	6527.5	0%	9,144.51	52.75
22	am13f	13	7699.5	7836.5	2%	11,920.70	53.53
23	14a	14	2904	2921	1%	22,389.72	56.42
24	14b	14	2736	2763	1%	9,151.56	57.71
25	P15	15	3195	3221	1%	89,923.20	61.99
26	P17	17	-	4865	-	-	71.67
27	P18	18	-	5665.5	-	-	75.70



(a) Instance S9 (b) Instance P18

Figure 5. The best obtained fitness of each generation

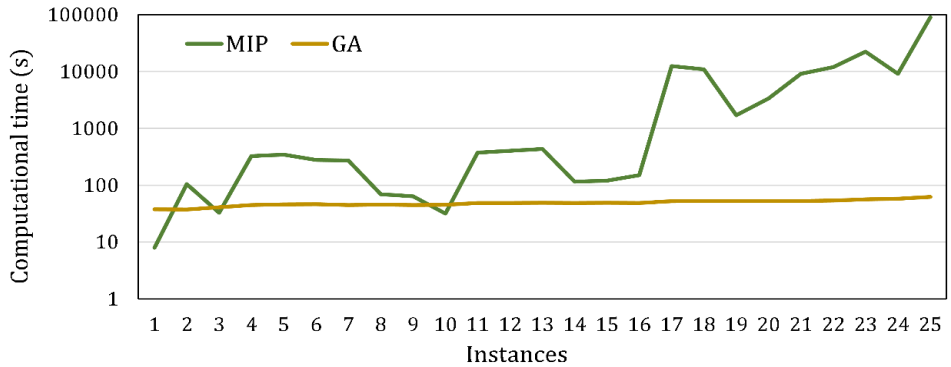


Figure 6. Comparison on computational time between GA and MIP

Conclusion

This paper presents a GA for solving the DRLP with aims to minimize the objective function of material handling operational cost in shorter computer time. A set of instances with various complexity are used to evaluate the performance of the proposed GA. The results show that the proposed GA can deliver good solutions with small deviation from the optimal value obtained by MIP. The advantage of the proposed GA is that it can provide the solution in much shorter time than the MIP. Thus, it is suitable for computation of instances with more complexity.

The future improvement can be explored by considering the exact position of the machine in the shorter row. In the current calculation, the first machines of both rows are placed from the same starting point. In the real application, the location of machines in the shorter row can be adjusted since this row has extra space, so that the material handling cost can be further minimized.

References

1. Mohamadghasemi, A., and Hadi-Vencheh, A., An Integrated Synthetic Value of Fuzzy Judgments and Nonlinear Programming Methodology for Ranking the Facility Layout Patterns, *Computers and Industrial Engineering*, 62(1), 2012, pp. 342-348.
2. Amaral, A. R., An Exact Approach to the One-dimensional Facility Layout Problem, *Operations Research*, 56(4), 2008, pp. 1026-1033.
3. Amaral, A. R., A Mixed-integer Programming Formulation for the Double Row Layout of Machines in Manufacturing Systems, *International Journal of Production Research*, 57(1), 2019, pp. 34-47.
4. Chung, J., and Tanchoco, J. M. A., The Double Row Layout Problem, *International Journal of Production Research*, 48(3), 2010, pp. 709-727.
5. Amaral, A. R., Optimal Solutions for the Double Row Layout Problem, *Optimization Letters*, 7(2), 2013, pp. 407-413.
6. Secchin, L. D., and Amaral, A. R., An Improved Mixed-integer Programming Model for the Double Row Layout of Facilities, *Optimization Letters*, 13(1), 2019, pp. 193-199.
7. Murray, C. C., Smith, A. E., and Zhang, Z., An Efficient Local Search Heuristic for the Double Row Layout Problem with Asymmetric Material Flow, *International Journal of Production Research*, 51(20), 2013, pp. 6129-6139.
8. Zuo, X., Murray, C. C., and Smith, A. E., Solving an Extended Double Row Layout Problem using Multiobjective Tabu Search and Linear Programming, *IEEE Transactions on Automation Science and Engineering*, 11(4), 2014, pp. 1122-1132.
9. Wang, S., Zuo, X., Liu, X., Zhao, X., and Li, J., Solving Dynamic Double Row Layout Problem via Combining Simulated Annealing and Mathematical Programming. *Applied Soft Computing*, 37, 2015, pp. 303-310.
10. Amaral, A. R. S., A Heuristic Approach for the Double Row Layout Problem, *Annals of Operations Research*, 2020, pp. 1-36.
11. Deb, K., *Multi-objective Optimization using Evolutionary Algorithms* (Vol. 16), John Wiley and Sons, 2001.
12. Goldberg, D. E. Genetic Algorithms in Search, *Optimization, and Machine Learning,* 1989.
13. Rifai, A. P., Dawal, S. Z. M., Zuhdi, A., Aoyama, H., and Case, K., Reentrant FMS Scheduling in Loop Layout with Consideration of Multi Loading-unloading Stations and Shortcuts, *The International Journal of Advanced Manufacturing Technology*, 82(9-12), 2016, pp. 1527-1545.
14. Datta, D., Amaral, A. R., and Figueira, J. R., Single Row Facility Layout Problem using a Permutation-based Genetic Algorithm, *European Journal of Operational Research*, 213(2), 2011, pp. 388-394.
15. Hou, S., Wen, H., Feng, S., Wang, H., and Li, Z., Application of Layered Coding Genetic Algorithm in Optimization of Unequal Area Production Facilities Layout, *Computational Intelligence and Neuroscience*, 2019.
16. Simmons, D. M., One-dimensional Space Allocation: An Ordering Algorithm. *Operations Research*, 17(5), 1969, pp. 812-826.
17. Amaral, A. R., On the Exact Solution of a Facility Layout Problem. *European Journal of Operational Research*, 173(2), 2006, pp. 508-518.