American J. of Engineering and Applied Sciences 3 (1): 15-24, 2010
ISSN 1941-7020

Application of Intelligence Based Genetic Algorithm for Job Sequencing Problem on Parallel Mixed-Model Assembly Line

A. Norozi, M.K.A. Ariffin and N. Ismail
Department of Mechanical and Manufacturing Engineering, University Putra Malaysia,
43400 UPM Serdang, Malaysia

Abstract: Problem statement: In the area of globalization the degree of competition in the market increased and many companies attempted to manufacture the products efficiently to overcome the challenges faced. **Approach:** Mixed model assembly line was able to provide continuous flow of material and flexibility with regard to model change. The problem under study attempted to describe the mathematical programming limitation for minimizing the overall make-span and balancing objective for set of parallel lines. **Results:** A proposed mixed-integer model only able to find the best job sequence in each line to meet the problem objectives for the given number of job allotted to each line. Hence using the proposed mathematical model for large size problem was time consuming and inefficient as so many job allocation values should be checked. This study presented an intelligence based genetic algorithm approach to optimize the considered problem objectives through reducing the problem complexity. A heuristic algorithm was introduced to generate the initial population for intelligence based genetic algorithm. Then, it started to find the best sequence of jobs for each line based on the generated population by heuristic algorithm. By this means, intelligence based genetic algorithm only concentrated on those initial populations that produce better solutions instead of probing the entire search space. **Conclusion/Recommendations:** The results obtained from intelligence based genetic algorithm were used as an initial point for fine-tuning by simulated annealing to increase the quality of solution. In order to check the capability of proposed algorithm, several experimentations on the set of problems were done. As the total objective values in most of problems could not be improved by simulated algorithm, it proved the well performing of proposed intelligence based genetic algorithm in reaching the near optimal solutions.

Key words: Intelligent based genetic algorithm, simulated annealing, mixed model assembly line

INTRODUCTION

Evolutionary computing is a research area within computer science that used for solving combinatorial optimization and complex problems which they perform base on principles of generic population-based heuristic techniques (Eiben and Smith, 2003). With the emergence of meta-heuristic algorithms in recent years, so many complex problems have been studied and solved by metaheuristic search techniques such as Ant colony optimization, Tabu Search, Genetic Algorithm and Simulated Annealing have been employed to deal with complex scheduling problems. Many Meta-heuristic algorithms were applied to overcome the complexity of sequencing problems in assembly lines problems. Genetic algorithm is introduced by Goldberg (1989) as it works based on the procedure of natural mechanism and natural genetic. The population is composed of a collection of chromosomes which each string is encoded the problem solution as a finite-length of gens. The entire evolution process works based on natural mechanism. Evolutionary computing algorithms usually reach to the good solutions in the reasonable amount of time though the achieved solution can be local or global optimum. A two-stage flow shop problems is considered by Johnson (1954) and the proposed heuristic algorithm was developed to minimize the completion time. By increasing the complexity of practical problems in real world, sequence-dependent setup times become one of the most favored assumptions in the area of scheduling researches (Naderi, *et al*. 2008). A mixed integer programming model is developed by Wagner which minimizes the makespan in permutation flow shop

Corresponding Author: A. Norozi, Department of Mechanical and Manufacturing, Engineering University Putra Malaysia, 43400 UPM Serdang, Malaysia

(F_m/permu/C_{max}) with an arbitrary number of machines. This formulation focuses on minimizing the total idle time on the last machine which is associated with minimizing the total idle time on the last machine (Naderi, 2002). A comparison of two metaheuristic search in flow-line manufacturing cell was done by (Skorin-Kapov and Vakharia, 1993). The problem focuses on sequencing the part families with similar setup time and the heuristic search techniques was developed based on Tabu search. The results prove the outperforming of developed search techniques in comparison to previous simulated annealing. A hybrid Simulated Annealing and Tabu search is introduced by (Lin and Ying, 2009) for scheduling the non-permutation flowshop problems. The objective considered in this problem was focus on optimizing the make-span time for non-permutation flowshop scheduling. The performance of hybrid search algorithm was compared to several metaheuristic algorithms such as Tabu Search, ant colony optimization and simulated annealing and the results confirm the well performance of hybrid approach.

Problem statement: The problem under study attempts to describe the mathematical programming limitation for solving set of parallel lines. A mixed-integer model was developed by Wagner (Pinedo, 2002) to find the best job sequence that minimize the make-span for a single line. The number of jobs assigned to each line is predetermined for a single line problem, while for set of parallel lines different number of jobs can be assigned to each line that minimizes the overall make-span time of system so the mathematical model should be formulated for different values of job allocated to every single line. Meanwhile this study presented an intelligence based search approach to address the job allocation problem for parallel mixed-model assembly line to minimize the overall makespan and also balance the lines in way that all lines have almost equal processing time. A simple evolutionary based algorithms like GA, SA or etc also faced with difficulties as there is no guaranty that which configuration of job allocation provides best solution so an intelligence based genetic algorithm is developed to decrease the problem complexity through providing some degree of proficiency in selecting the potential . This study is organized as follows: A description of problem under study is provided to clarify the problem's assumptions and the mathematical programming model for problem under study is described. The complexity of problem under study is discussed to demonstrate understanding of research problem and an intelligence based genetic algorithm is applied to overcome the problem complexity. A numerical example is illustrated in the following. Conclusion and future research are presented in the last part.

MATERIALS AND METHODS

Typically, a mixed model assembly line is equipped with flexible workstations which are capable of producing variety of product models similar in product characteristics continuously and concurrently (Groover, 2001). The problem under study includes a number of parallel mixed-model assembly lines and each line consists of number of workstations which are capable of serving any job. A workstation in any line should be setup for the new materials requirement to be able to serve the new set of products. Initial setup time is essential for the first job of sequence and change over time is required to change the settings between jobs in the same line. The following assumptions are considered in this research: All the assembly lines perform assembly operation independently. The workstation time for every single job at all workstations are specified. Once the job allocated to any line, jobs are not allowed to shift to other assembly lines. No line should be left without job assignment. Each assembly line represents type of flow shop system and the workstations representative of involved machines in the flow shop system and in a larger prospect, the whole system likes parallel machine scheduling problems. Each assembly line acts as flow line system in which the overall make-spam for set of parallel lines is determined by the longest completion of line so minimizing the completion time of all lines directly effect on overall make span of system. Model diagram of problem under study is illustrated in Fig. 1.

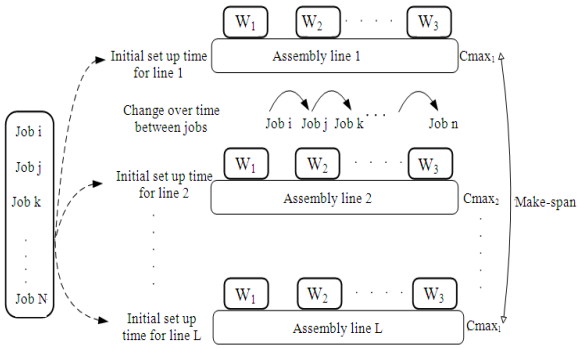

Fig. 1: Model diagram of parallel lines

Mathematical programming for job allocation: The proposed mixed-integer programming model is built based on flow shop model which is developed by Wagner (Pinedo, 2002) and it is expanded to consider the effect of initial setup time and sequential change over time for multiple lines. The mathematic formula is as follow:

$$\text{Min}\left\{(\text{Max}\{C\max_l\}) + \sum_{e=1}^{L-1}\sum_{k=e+1}^{L} B_{ek}\right\} \quad (1)$$

Subject to:

$$\text{Idle time}_l = \sum_{i=1}^{m-1}\sum_{j=1}^{n}(X_{j,1,l} * p_{ij}) + \sum_{j=1}^{k_l-1} l_{m,j,l}\ ;\ L = 1, \ldots, L \quad (2)$$

$$\text{Setup}_l = \sum_{k=1}^{k_l-1}\sum_{i=1}^{n}\sum_{j\neq i=1}^{n}(C_{ij} + S_{ik}) * (X_{i,k_l,l}) * (X_{j,k_l+1,l}); \quad l = 1,\ldots L \quad (3)$$

$$\text{Process}_l = \sum_{j=1}^{n}\sum_{k_l=1}^{k_l} X_{j,k_l,l} * P_{mj}; l = 1,\ldots L \quad (4)$$

$$C\max_1 = \text{Setup}_l + \text{Idle time}_1 + \text{Process}_l;\ l = 1,\ldots L \quad (5)$$

$$B_{ek} = \sum_{k=e+1}^{L} \left|T_e - T_k\right|; e = l,\ldots L-1 \quad (6)$$

$$T_l = \sum_{j=1}^{n}\sum_{k_l=1}^{k_l} X_{j,k_l,l} * t_j; L = l,\ldots L \quad (7)$$

$$\sum_{j=1}^{n} X_{j,k_l,l} = 1; k_l = 1,\ldots k_l, l = 1,\ldots L \quad (8)$$

$$\sum_{k_l=1}^{k_l} X_{j,k_l,l} \leq 1; j = 1,\ldots n,\ l = 1,\ldots L \quad (9)$$

$$\sum_{L=1}^{L}\sum_{k_l=1}^{k_l} X_{j,k_l,l} = 1; j = 1,\ldots n \quad (10)$$

$$l_{i,k_l,l} + \sum_{j=1}^{n} X_{j,k_l+1,l} * P_{ij} + W_{i,k_l+1,l} - W_{i,k_l,l} - \sum_{j=1}^{n} X_{j,k_l,l} * P_{i+1,j} - l_{i+1,k_l,l} = 0; K = 1,\ldots, k_l - 1, i = 1,\ldots\ldots m-1, l = 1,\ldots L \quad (11)$$

$$W_i 1l = 0;\ i = 1,\ldots \ldots m\text{-}1,\ l = 1,\ldots L \quad (12)$$

$$L_{1k,l} = 0;\ k = 1, \ldots \ldots n\text{-}1,\ l = 1,\ldots L \quad (13)$$

$$S_{ik} = 0;\ k = 2 \ldots\ldots n\text{-}1 \quad (14)$$

$W \geq 0,\ I \geq 0$

$X_{ijk} \in 0,1$

X_{jkl} = If job j is the kth job in the sequence in line L

I_{ikl} = Ideal time on machine i between the processing of job in the kth and (k+1)th position in assembly line L

W_{ikl} = Waiting time of the job in the kth position in between machine i and i+1 in the lth assembly line

m = Number of workstations in assembly line

n = Number of jobs in flow shop system

S_{ik} = Initial setup time for job i in the kth position of job sequence

C_{ij} = Change over time between job i and j

Idle time_l = Total idle time at the last workstation for lth assembly line.

Setup_l = Total setup time for lth assembly line

$C\max_l$ = Completion time for lth assembly line

Process_l = Processing time at the last workstation of lth assembly line

B_{ek} = Total absolute difference among process time of line e and rest of lines

t_j = Total Process time for job j

T_i = Total process time of lth assembly line

k_l = Number of jobs allocated to lth assembly line

L = Number of assembly line

P_{ij} = Process time of job j at workstation i

P_{mj} = Process time of job j at workstation m

The first term of objective function (1) attempts to minimize the overall make-span of this system is by minimizing the longest completion time of lines. The second term of objective function attempts to balance workload among all assembly lines by considering all jobs' process time for every single job. Minimizing the absolute value of total differences in process time of all assembly lines is the procedure that is used for achieving this goal.

Constraint explanation: Minimizing the makespan time in (F_m/permu/C_{max}) is associated with minimizing the total idle time on the last workstation. The second set of Equation 2 is used to obtain the minimum total idle time at the last workstation for every single assembly line. Equation 3 calculates the total setup time by summing the initial setup time and the change over time between the different jobs in sequence for each

assembly line. It is obvious that initial setup time is only considered for the first job of sequence. Equation 4 determines the processing time at the last workstation for every assembly line. Generally in the simple flow shop system, the completion time is achieved by summing the process time and idle time on the last workstation of the corresponding line. Equation 5 calculates the total completion time for every single assembly line by adding job's initial setup time and change over time to the flow time of the corresponding line. Equation 6 helps to find the difference in total process time for multiple lines by computing the absolute difference in total process time for lines. Equation 7 is used to attain the process time for every single line which is determined by summing all the jobs allocated to the assembly lines. The total process time for every single assembly line is attained by summing the process times of all the jobs allocated to that assembly line. Constraint (8) is used to dedicate all jobs to the available positions in which each job is placed at the unique position of that assembly line. Constraint (9) ensures that each job can be placed in only one of the available positions of sequence for each assembly line. Constraint (10) ensures that from all the available positions in the system, each job must be processed in only one of all available positions of sequences. The last set of constraints 11 show the inevitable relation between the idle time and waiting time in each assembly line. It represents the logical concept of involve variables in flow shop system. Equation 12 reveals that the waiting time for the first job in a sequence is always equal to zero for any assembly line. Equation 13 shows that the first workstation is always ready to process the first job of a sequence in any assembly line. Equation 14 illustrates that the initial setup time is only considered for the first job of sequence and for the rest of jobs is zero.

Complexity theory: The proposed mixed-integer model is able to find the best job sequence in each line to meet the problem objectives for a predetermined value of k_l. specifying the best value of k_l that provides opportunity for mixed integer model to be solved by exact methods are quite time consuming and inefficient while the problem should be formulated for all possible value of k_l. The total permutation of job allocation to set of parallel lines can be computed as follows:

$$\text{Total permutation} = \sum_{C=1}^{N_c} n! \quad (15)$$

Where:

N_c = Set of configurations for job allocation to assembly lines
N = Number of jobs

It should be noted that the different number of jobs can be allocated to each line so the total permutations for this problem is obtained by summing the all possible configurations of job allocation. Due to massive permutations of job allocation, intelligence based genetic algorithm is applied to probe the solution space to find the near optimal solutions through reducing searching space by choosing the set of potential members of N_c that provides best job allocation and sequence that meet both problem objectives. In this case, the problem complexity is tended to n!.

Problem solving procedure: All jobs involved in the system can be assigned to set of parallel assembly line in different way in which the total summation of all allocated jobs are fixed. This process provides different configuration of job allocation which increases the complexity of parallel mixed assembly line problems so different number of jobs can be allocated to each line, while only one of them can provides potential situation that may conduce to the best sequence of allocated jobs to meet the presumed objectives. The proposed mixed-integer model gives the optimum solution for the given configuration of job allocation for each line while checking all the possible value of k_1 for large size problems requires huge problem formulation and massive computation by exact methods. A simple genetic algorithm also faces with difficulties as there is no guaranty that which configuration of job allocation provides the best solution, because it's directly associated with job's process time so all configuration of job allocation should be checked. As can be seen from Eq. 15, this value can dramatically increases in large size problem, so checking all the configurations of job allocation would be so time consuming. In this case, an intelligence based evolutionary algorithm should be applied to solve the problem and find the solutions in efficient way. In a usual genetic algorithm, the fitness function is a particular function which quantifies the quality of generated chromosome and these functions are usually predetermined and specified with regard to objectives of the corresponding problem and they are usually expressed in terms of mathematical equations or even set of rules. As in each configuration of job allocation, several permutations of job allocation and job sequence are available so in the proposed intelligence based genetic algorithm, GA-2 is executed as a cost evaluation function to find the minimum

attainable cost for the corresponding chromosome. In this regard, the fitness function for GA-1 is not a specified function but the best result that can be achieved by GA-2 is considered as a related cost. The total configurations of job allocation are obtained by solving the Eq. 16:

$$\begin{cases} l_1 + l_2 + \ldots + l_1 = n \\ l_1 \geq l_2 \geq \ldots \geq l_1,\ l_1 \in \text{Integer} \end{cases} \quad (16)$$

Where:

l_l = Represents number of jobs assigns to the lth assembly line

n = Shows the total number of jobs in the system

The best configuration of job allocation is directly depends on jobs' process time so all should be checked to find the best solution. In order to tackle the complexity of this problem, an intelligence based genetic algorithm is proposed which provides some degree of intelligence to select the best configuration of job allocation and let the GA-2 to find the best sequence of allocated job based on the given job allocation configuration to avoid checking all the potential solution which is so time consuming. As illustrated in Fig. 2, a population of potential solution which can meet the condition (Eq. 16) is randomly generated. The cost computation for each chromosome is done by GA-2 through finding the best sequence of jobs which is accompanied with the best job allocation.

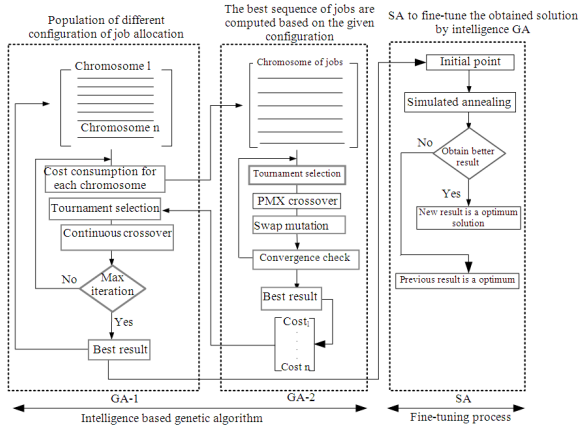

Fig. 2: Diagram of intelligence based genetic algorithm and fine-tuning process

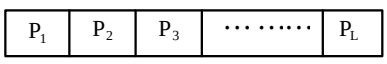

Fig. 3: Chromosomes of integer numbers for GA-1

The chromosomes are sorted according to their cost value and proceed to the genetic operators in (GA-1) for further evolution. In this process, GA-1 attempts to find the potential configuration of job allocation that has higher probability for better job allocation and sequencing. Meanwhile this probability is computed by GA-2 through finding the optimum job allocation and job sequence that optimize the presumed objective functions. This process continues until the optimum value is achieved. In the final step, a simulated annealing is used to fine-tune the best results obtained by intelligence based genetic algorithm to improve the quality of solutions. GA-1 aims to intelligently decrease the N_c and chose those configurations of job allocation that there is a higher possibility of optimum solution. It lets the GA-2 to mainly focus on specific value of N_c which is directly conduced to a better near optimal solution through reducing the search space. The flowchart of intelligence based GA and fine-tuning is shown in Fig. 2.

As mentioned in problem solving section, GA-1 aims to find the best configuration of job allocation by GA-2 for finding the optimum job sequence that minimizes the objective function. The possible solution for Equation 16 can be set of integer value between 1 to n as it satisfies the condition. A chromosome for GA-1 is a string of length L where it is composed of some integer values as the total summation of distributed jobs among lines is equal to total number of jobs in system. The chromosome for GA-1 is shown in Fig. 3.

Where:

$$\sum_{l=1}^{L} P_l = n \quad (17)$$

P_l = Represents number of jobs assigns to the lth assembly line

n = Shows the total number of jobs in the system

Example: In order to distribute 10 jobs among 4 lines, the chromosome can be initiated as follows: [4, 3, 2, 1] or [5, 2, 2, 1] or [3, 3, 2, 2].

Initial population generation: In order to generate the initial population for GA-1, a simple heuristic algorithm is proposed to generate possible solutions for different configuration of job allocation which as follows:

- The following Linear Programming (LP) models capable of determining the upper and lower bounds of possible solutions for distributing n jobs among l

assembly line. Lp1 aims to maximize the maximum value of any involved variables as much as possible so it only force one of the variables to reach to the maximum while the rest of variables get the minimum value. The value obtained form LP1 is called upper bound for job allocation which guaranty that no lines is left without job assignment:

$$\begin{aligned} &\text{LP1}\\ &\text{Max}\{\max\{x_l\}\};\ l = 1,..., L\\ &\sum_{i=1}^{L} x_l = n\\ &x_l \geq x_{l+1};\ l = 1,\ldots L\\ &x_l \geq l; l = 1,\ldots L\\ &x_l \in \text{Integer} \end{aligned} \tag{18}$$

- LP2 attempts to maximize the minimum value of all variables so all intend to be in a minimum value difference. The solution achieved by LP2 is called lower bound:

$$\begin{aligned} &\text{LP2}\\ &\text{Max}\{\max\{x_l\}\};\ l = 1, ..., L\\ &\sum_{i=1}^{L} x_l = n\\ &x_l \geq x_{l+1};\ l = 1,\ldots L\\ &x_l \geq l; l = 1,\ldots L\\ &x_l \in \text{Integer} \end{aligned} \tag{19}$$

- In order to generate new solutions, a particular combining technique is applied to construct new data points within the range of upper and lower point. The equation (20) and (21) are able to generate new points where α and β are arbitrary coefficient:

$$\begin{cases} Z_l = [\alpha * (X_l) + (1-\alpha) * (Y_l);\ l = 2,\ldots, L & (20)\\ z_1 = n - \sum_{l=2}^{L} Z_l & (21) \end{cases}$$

Where:

α = Random number on the interval [0, 1]
X_l = lth variable in the upper bound set
Y_l = lth variable in the lower bound set
Z_l = lth variable in the new generated point

- A set of new generated points provide an initial population for GA-1 to start the intelligence based genetic algorithm

An example is provided to clarify the implemented techniques for generating new points. Twenty jobs are assigned to 4 lines in which each line must serve at least one job:

$$\begin{cases} l_1 + l_2 + l_3 + l_4 = 20\\ l_1 \geq l_2 \geq l_3 \geq l_4, l_1 \in \text{Integer} \end{cases}$$

The solution obtained by LP1 and LP2 are illustrated as follows:

LP1 = Upper bound = [$x_1 = 17$, $x_2 = 1$, $x_3 = 1$, $x_4 = 1$]
LP2 = Lower bound = [$y_1 = 5$, $y_2 = 5$, $y_3 = 5$, $y_4 = 5$]

For a given $\alpha = 0.3$, a new generated point is $Z_2 = [0.3*(1)+(1-0.3)*(5)]$; $l = 2,\ldots,4$ then [$Z_2 = 4$, $Z_3 = 4$, $Z_4 = 4$] and finally $Z_1 = 20 - \sum_{l=2}^{4} Z_1 = 8$. Meanwhile the new generated point is [$Z_1 = 8$, $Z_2 = 4$, $Z_3 = 4$, $Z_4 = 4$] which is produced by linear interpolating of n-1 variables. The first variable is not engaged in interpolation process to keep the number of jobs fix during the whole generation process. A population of new points can be produced by generating a random value of α.

Crossover operator: As the entire algorithm moves forward those generated points that have minimum value of cost function have higher probability to be a part of candidate region around the optimum point so continuous crossover capable of producing new offspring inside the candidate region to do further evolution as the generation moves on. Meanwhile, a new offspring can be produced while carrying the information from both parents. The blending methods for this problem can be done by finding ways to combine variable values from the two parents into new variable values while keeping the jobs number fixed during the crossover process. A single offspring variable value comes from a combination of the n-1 variables of two corresponding parents' variable. Producing new offspring can be done through generating two different random value of β and combining the selected parents. The entire crossover procedure is shown as follows:

$$\begin{cases} C_1 = [\beta * (P1_l) + (1-\beta) * (P2_l)];\ l = 2,\ldots L & (22)\\ c_1 = n - \sum_{l=2}^{L} c_1 & (23) \end{cases}$$

Where:
β = Random number on the interval [0, 1]
$P1_l$ = lth variable in the mother chromosome
$P2_l$ = lth variable in the father chromosome
C_1 = lth variable in the new generated point
n = Number of jobs

In the continuous crossover, some gens are randomly selected to be combined while for the proposed crossover operator, the combining process should be done for all the n-1 variables and it can be seen as a main difference between ordinary continuous crossover and the proposed crossover operator. Figure 4 illustrates the proposed crossover operator for the considered problem. Crossover operator is considered as a main genetic operator in genetic algorithm so mutation operator is not executed in GA-1.

GA-2 for Job allocation problem: GA-2 attempts to allocate jobs to the assembly lines and find the best job sequence in order to minimize the objective functions. As the best order of jobs provide the optimum solution, it falls to permutation based genetic algorithm category. The chromosome is a string of length N where k_{lc} ; l = 1,… L represents the number of jobs assigns to the lth assembly line in the C^{th} configuration of job allocation. Figure 5 represents a chromosome of tasks and the shows how they are assigned to the workstations.

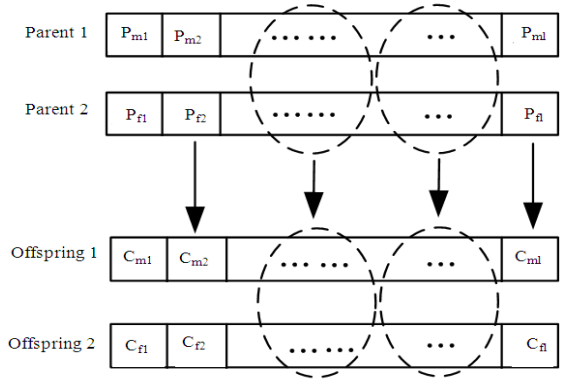

Fig. 4: Continuous cross over operator for GA-1

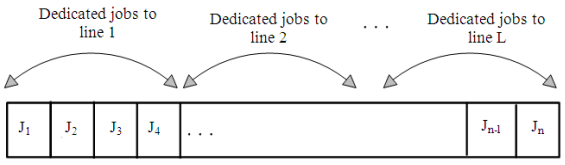

Fig. 5: Chromosome of jobs

Generally an appropriate fitness function closely associates with mathematical objective function which is capable of computing the cost for each chromosome quickly. Fitness function is used to evaluate the generated chromosomes to measure the optimality of solutions. Total objective value is computed by summing the value of make-span time, process time difference and completion time difference. The proposed fitness function is given by:

$$F = 1/\left(\max\{C\max_l\} + \sum_{e=1}^{L-1} \sum_{k=e+1}^{L} \left| T_e - T_k \right| \right)^2 \quad (24)$$

Where:
$Cmax_l$ = Completion time for lth assembly line
T_k = Total process time of kth assembly line
L = Number of assembly lines

Those chromosomes that provide minimum make-span time and also balance the lines are selected for mating operation. This process continues until GA-2 finds the best job sequence and allocation for the corresponding chromosome of GA-1. In the next step, the second chromosome of GA-1 is selected and the best attainable cost is computed through GA-2. All the computed costs for the entire population members are transferred to GA-1 for sorting operation.

Tournament selection: Tournament selection is a very popular strategy that aims to imitate natural competition of specious (Michalewicz, 1996). The tournament selection works in the way that two individuals are randomly selected from the mating pool. The individual with the highest fitness value is selected as the winner of the tournament and the selection process continues by selecting a new tournament group randomly until all the individuals are selected. Finally the winner of each competition is copied to the worst chromosomes. Tournament selection is applied in both GAs as selecting mechanism for choosing the best individuals within population. Elitism is usually used to prevent the loss of the current fittest member of the population due to crossover or mutation operators and keep the best individual from generation to generation (Haupt and Haupt, 1997). Elitism is applied through genetic programming.

Genetic parameter setting: Genetic Algorithm parameter setting aims to increase the algorithm performance by setting the genetic parameters by optimal values. The initial population is composed of a set of individuals, which are generated by using random generator. The size of population for both GA-1 and

GA-2 are fixed during all generations. Crossover is considered as most important genetic operator which combines set of information from different chromosomes and generates new offspring which captures the both individuals information. Partially Mapped Crossover (PMX) is employed as crossover operator in GA-2 algorithm which the cross over rate is set based on initial population size. Initial population size is directly associated with providing more diversity of potential solution which is varied for different complexity of problem. The complexity of job allocation problem is increasing by n! order so an appropriate level of population is required to provide more diversity of potential solutions and discourages premature convergence to local optimums. The population size for GA-2 is set to 80 with 50% of cross over rate which is used by many researchers and able to find good solution in a reasonable amount of time (Grefenstette, 1986). Mutation operator aims to provide a means to prevent algorithm from rapid convergence or premature convergence and drive algorithm to search further feasible problem space to escape from local optimum. For this means swap mutation is elected as mutation operator. The Mutation probability is set to 0.02 in GA-2 algorithms which is a typical value for Genetic Algorithm (Leu, *et al*. 1994). Total number of generation is used as a stopping criterion in GA-2 program to terminate the algorithm at 300 generations.

Simulated annealing-fine-tuning: Simulated annealing is able to deal with noisy search space and complex problems. In the annealing process, the temperature of the molten metal decreases until the crystal is frozen. If the cooling procedure is done quickly some structural irregularities will happen in the atomic structure. The algorithm starts with a small random perturbation to the atomic structure. If this results in the lower energy sate, the algorithm is repeated by using new energy state. But if the higher energy state is achieved through the new atomic structure, the new state is accepted with certain probability which is depends on the history of the search (Winston, 2003). Simulated annealing is only used to fine-tune the solution obtained by intelligence based genetic algorithm to improve the quality of solutions. This procedure helps to find the optimum solution if it was not found in previous step. It also confirms the well performing of proposed search algorithm when no improvement is achieved during the fine-tuning algorithm. For each problem, the best job sequence which is obtained by intelligence based genetic algorithm is used as an initial point for simulated annealing. A neighborhood search is used as a main operator for exploring different solutions. Neighborhood search generates a new atomic structure by changing the candidate solution in order to visit more potential solutions within the search space. In this case, two jobs are randomly swapped by generating two random keys (Naderi, *et al*. 2008). In order to avoid algorithm to reach to the local optimum, some worse moves might be accepted based on current temperature. The exponential cooling scheduling is used in this research as it believed to be an appropriate cooling schedule for the SA (Wang and Zheng, 2001). The initial experiment demonstrated us that the temperature over the range 40-50 is proper for fine-tuning process and the stopping temperature is fixed at 0 while cooling temperature is set to 0.05.

RESULTS AND DISCUSSION

In order to check the efficiency of proposed procedure, different numbers of jobs are allocated to the lines which each considered as a new problem that should be solved by intelligence based genetic algorithm. There are three lines in which each consists of two workstations. The first problem starts with the first 10 jobs in the system and the problem complexity is rising as the number of jobs increasing until reach to the maximum of 15. For each problem, upper bound and lower bound is computed by LP1 and LP2 to determine the range of variation for chromosomes of GA-1. Table 1 illustrates the required process time and the amount of workload in workstations for every single job. Table 2 includes the initial setup time and change over time matrix for all jobs. Metaheuristic algorithms only guaranty the local optimality so the best solution that provides minimum objective value is selected as a near optimal solution. Different experiments are done based on different number of jobs and for each problem, the chromosome values range between the lower bound and the upper bound within the problem.

Table 1: Job process time and workload at workstation

		Work load	
Job	Process time	W1	W2
1	160	80	80
2	40	30	10
3	30	18	12
4	240	120	120
5	700	450	250
6	200	100	100
7	300	150	150
8	840	420	420
9	100	70	30
10	150	75	75
11	180	100	80
12	350	200	150
13	200	100	100
14	1500	650	850
15	2000	1200	800

Table 2: Initial setup time and changeover time

	J1	J2	J3	J4	J5	J6	J7	J8	J9	J10	J11	J12	J13	J14	J15	Initial setup time
J1	0	12	8	7	11	13	5	20	19	16	20	20	10	25	15	40
J2	20	0	20	18	13	18	16	20	10	8	5	15	10	16	43	50
J3	19	10	0	20	10	12	7	28	15	10	5	15	20	8	56	60
J4	18	12	10	0	10	5	20	18	15	12	18	16	10	9	18	100
J5	6	9	12	9	0	20	25	13	34	23	45	13	40	26	32	140
J6	20	9	12	10	20	0	6	50	30	42	20	15	20	32	14	30
J7	20	12	10	20	30	45	0	18	10	20	35	20	20	14	19	40
J8	30	18	12	30	60	10	30	0	12	18	20	20	20	10	22	150
J9	10	15	16	20	20	30	18	45	0	24	34	33	13	16	33	30
J10	20	30	45	34	20	10	28	32	45	0	30	20	24	32	43	20
J 11	25	35	40	30	20	15	25	32	25	20	0	24	42	28	13	120
J 12	12	42	33	34	23	14	15	32	26	35	44	0	34	12	17	50
J 13	12	23	34	35	42	21	23	42	34	23	43	23	0	16	20	40
J 14	22	42	33	34	23	14	15	32	26	35	44	34	43	0	10	200
J 15	55	34	29	30	80	73	43	65	39	18	20	15	11	9	0	300

Table 3: Intelligent based genetic algorithm

Problem	No. of job	Upper bound	Lower bound	Selected configuration of job allocation	Make-span	Time difference	Objective value
1	10	[8, 1, 1]	[4, 3, 3]	[3, 4, 3]	964	100	1064
2	11	[9, 1, 1]	[4, 4, 3]	[6, 2, 3]	967	40	1007
3	12	[10, 1, 1]	[4, 4, 4]	[4, 3, 5]	1050	20	1070
4	13	[11, 1, 1]	[5, 5, 3]	[4, 6, 3]	1067	20	1087
5	14	[12, 1, 1]	[5, 5, 4]	[6, 6, 2]	1645	20	1665
6	15	[13, 1, 1]	[5, 5, 5]	[5, 6, 4]	2274	0	2274

Table 4: Job sequence for every single line

No. of job	Line 1	Line 2	Line 3
10	2 8 3	7 4 1 6	10 5 9
11	3 6 7 4 2 1	10 8	9 11 5
12	10 2 6 5	9 1 8	7 3 4 12 11
13	13 5 1 9	3 7 4 2 6 12	10 8 11
14	10 2 4 13 8 6	5 9 7 11 12 3	1 14
15	11 14 10 7 6	1 5 8 12 2 4	9 13 15 3

Table 5: Fine-tuning process by SA

Problem	No. of job	Selected configuration of job allocation	Make-span	Time difference	Objective value
1	10	[4, 3, 3]	964	100	1064
2	11	[6, 3, 2]	967	40	1007
3	12	[5, 4, 3]	1009	20	1029
4	13	[6, 4, 3]	1067	20	1087
5	14	[6, 6, 2]	1645	20	1665
6	15	[6, 5, 4]	2230	0	2230

Table 6: Final result for job sequence for every single line

No. of job	Line 1	Line 2	Line 3
10	2 8 3	7 4 1 6	10 5 9
11	3 6 7 4 2 1	10 8	9 11 5
12	4 11 12 7 3	1 2 6 5	10 8 9
13	13 5 1 9	3 7 4 2 6 12	10 8 11
14	10 2 4 13 8 6	5 9 7 11 12 3	1 14
15	7 11 1 10 2 14	4 5 12 8 13	6 15 3 9

Table 3 shows the result obtained by intelligent based genetic algorithm in which the total objective value is computed by summing the make-span time and time difference between the lines which is shown in Table 3. The corresponding job sequence for each problem is shown in Table 4. As can be seen from the results, there is no time difference between the lines in problem 6 and also this value is reach to the minimum of 20 in problem 3, 4 and 5. The selected configuration of job allocation for each problem clearly prove that there is no rules to determine the best configuration of job allocation as it directly based on jobs process time even though this value may dramatically change with a little changes in jobs time even with the same number of job. The solution obtained by intelligence based genetic algorithm is used as an initial point for fine-tuning process. The solution obtained from fine-tuning process is illustrated in Table 6. As can be seen from Table 6, no improvement is achieved by SA in minimizing the second objective (time differences between lines) in all problems. It clearly confirms that the solution obtained by intelligence based genetic algorithm was optimum as the total objective value for problem 1-4 are fixed and no further improvements are gained by SA. Therefore the intelligence based genetic algorithm is straightly directed to the optimum solution in problem 1-4 in all objectives. Fine-tuning process helps to find better solutions in problem 3 and 6 in finding the better job sequences which results in improving the first objective function (overall make-span). Although the time differences in both problem 3 and 6 are fixed, utilizing fine-tuning algorithm leads to a shorter make-span form 1050 to 1009 in problem 3 and 2274 to 2230 in problem 6.

CONCLUSION

In this study, an intelligence based genetic algorithm is applied to tackle the complexity of

sequencing problem in parallel mixed-model assembly line problems. For solving such problems by mathematical methods the proposed multi objective mixed-integer model should be formulated for several configuration of job allocation which is quite time consuming and inefficient. A simple genetic algorithm also faces difficulties due to massive search space so the proposed search technique is implemented to reduce the problem complexity and overcome the required massive search space. The solving procedure initiates by generating initial population for different configuration of job allocation. The cost evaluation for the involved chromosomes is done by GA-2 and then the population is sorted according to computed cost. GA-2 tries to allocate jobs to the assembly lines in order to minimize the multi objective functions. The simulated annealing is applied to fine-tune the obtain solution in order to increase the quality of solutions. The achieved results from SA proves that proposed algorithm capable of finding the best sequence of allocated job in the most of problems. However, there are enormous opportunities for future work of this research by engaging more practical issues of material handling systems in order to feed the workstations which are widely used in many industry units. Meanwhile some new parameters and constraints are required to represent the system properties. Other systematic local search algorithms can be developed to reduce the problem complexity as well as increasing the solution quality.

REFERENCES

Eiben, A.E. and J.E. Smith, 2003. Introduction to Evolutionary Computing. 1st Edn., Springer, ISBN: 978-3-540-40184-1, pp: 299.

Goldberg, D.E., 1989. Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley Longman Publishing Co., Inc., Boston, MA., USA., ISBN: 0-201-15767-5, pp: 4412.

Grefenstette, J.J., 1986. Optimization of control parameters for genetic algorithms. IEEE Transactions Syst. Man Cybernet., 16: 122-128. DOI: 10.1109/TSMC.1986.289288

Groover, M.P., 2001. Automation, Production Systems and Computer-Integrated Manufacturing. 2dn Edn., Prentice Hall, London, NJ., ISBN: 10: 0130889784, pp: 856.

Haupt, R.L. and S.E. Haupt, 1997. Practical Genetic Algorithms. Wiley-Interscience, New York, ISBN: 10: 0471188735, pp: 192.

Johnson, S.M., 1954. Optimal two-and three-stage production schedules with setup times included. Naval Res. Logist. Q., 1: 61-68. DOI: 10.1002/nav.3800010110

Leu, Y.Y., L.A. Matheson and L.P. Rees, 1994. Assembly line balancing using genetic algorithms with heuristic-generated initial populations and multiple evaluation criteria. Dec. Sci., 25: 581-605. DOI: 10.1111/j.1540-5915.1994.tb01861.x

Lin, S.W. and K.C. Ying, 2009. Applying a hybrid simulated annealing and Tabu search approach to non-permutation flowshop scheduling problems. Int. J. Prod. Res., 47: 1411-1424. DOI: 10.1080/00207540701484939

Michalewicz, Z., 1996. Genetic Algorithms + Data Structures = Evolution Programs. 3rd Edn., Springer, ISBN: 10: 3540606769, pp: 387.

Naderi, B., M. Zandieh, A. Khaleghi Ghoshe Balagh and V. Roshanaei, 2008. An improved simulated annealing for hybrid flowshops with sequence-dependent setup and transportation times to minimize total completion time and total tardiness. Expert Syst. With Appli., 36: 9625-9633. DOI: 10.1016/J.ESWA.2008.09.063

Pinedo, M.L., 2002. Scheduling: Theory, Algorithms and Systems. Prentice Hall, Upper Saddle River, New Jersey, pp: 586.

Skorin-Kapov, J. and A.J. Vakharia, 1993. Scheduling a flow-line manufacturing cell: A tabu search approach. Int. J. Prod.. Res., 31: 1721-1734. http://cat.inist.fr/?aModele=afficheN&cpsidt=4781237

Wang, L. and D.Z. Zheng, 2001. An effective hybrid optimization strategy for job-shop scheduling problems. Comput. Res., 28: 585-596. DOI: 10.1016/S0305-0548(99)00137-9

Winston, W.L., 2003. Introduction to Mathematical Programming: Applications and Algorithms. Duxbury Resource Center, ISBN: 0534423574.