Scientific Research and Essays Vol. 6(8), pp. 1720-1731, 18 April, 2011
Available online at http://www.academicjournals.org/SRE
DOI: 10.5897/SRE09.441
ISSN 1992-2248 ©2011 Academic Journals

Full Length Research Paper

An optimization technique using hybrid GA-SA algorithm for multi-objective scheduling problem

A. Norozi*, M. K. A. Ariffin, N. Ismail and F. Mustapha

Department of Mechanical and Manufacturing Engineering, Universiti Putra, Malaysia, 43400 UPM Serdang, Malaysia.

Accepted 3 December, 2010

A Mixed-model assembly line is widely employed to perform the assembly operation in industries and the time needed to release products to market is frequently considered by many researchers. However, providing an appropriate level of flexibility to meet customer demand variations is critical for companies survival in this competitive market. The problem of production planning in terms of sequencing various product model is studied here. A manufacturing system is presented to show the application of this problem. A proposed multi-objective function is given to minimize the overall make-span of a mixed-model assembly line, but with additional goals also considered, such as balancing the assembly line and minimizing the variation of completion time. We propose a solution aimed to solve the problem in successive stages. For each stage, a mathematical model formally describes the problem and the main difficulties faced are explained. Due to the high complexity of problem solving procedures by classical mathematic techniques, this paper presents a new approach of hybrid genetic algorithm-simulated annealing (GA-SA) implementation in order to meet the problem objectives. A proposed hybrid scheme is executed to overcome problem complexity and to meet the problem objectives. In order to check the efficiency of hybrid search techniques, a comparison is made between the results obtained by hybrid GA-SA and GA, and the comparison validates the effectiveness of the presented hybrid search technique.

Key words: Genetic algorithm, hybrid GA-SA (genetic algorithm-simulated annealing), mixed-integer programming, meta-heuristic algorithm.

INTRODUCTION

One of the most important issues for companies is to release their products into the markets earlier than other competitors to take more market share and stabilize the company's situation in the global competitive markets. Creating more profit enhance the company's survival chance in the economic crisis period and improve the flexibility to respond to key business issues. Quick response manufacturing is an operation strategy which helps companies to stand against today's challenges in the competitive markets by focusing and reducing the time required to accomplish various manufacturing activities (Stevenson, 2005). The flexible assembly lines provide special benefits for production and assembling industries by providing more level of flexibility in producing different types of product if it is well planned (Rehg, 1994). They also make industries more efficient to adjust the production requirements to the possible demand changes (Sule, 1997). Mixed-model assembly lines are kinds of manufacturing system that is appropriate and quickly respond to the customer's variable demands. It can be measured as quick response manufacturing of the category which is used to produce different types of products without suffering from large inventory costs. In recent years, with the emergence of meta-heuristic algorithms, such as ant colony, tabu search, genetic algorithm, simulated annealing, have been employed to deal with scheduling problems. Many meta-heuristic algorithms were applied to overcome the complexity of mixed-model assembly lines problems. Simaria and Vilarinho (2009) attempts to develop an optimization algorithm for balancing two-sided assembly

*Corresponding author. E-mail: alireza.norozi.en@gmail.com.

Abbreviations: SDST, Sequence dependent setup time; **MOSS,** multi objective scatter search; **PCB,** printed circuit board; **PCA,** principal component analysis; **PMX,** partially mapped crossover; **GA-SA,** genetic algorithm-simulated annealing.

lines which are usually used in the automobile industry, more than one worker, are simultaneously working on the two sides of jobs.

Kim et al. (1996) considered a new mixed-model assembly line with hybrid workstations type and sequence-dependent setup time. Their study focuses on optimizing three objectives such as minimizing the overall length of line, change over time and production rate variation. A new Immune algorithm was introduced by Zandieh et al. (2006) for scheduling a hybrid flow shop in which there is sequence dependent setup time (SDST) considerations within the problem. The actual data in real world's problems is not strictly predetermined and has fuzzy nature of data. Sakawa and Kubota (2000) focused on the job shop scheduling with fuzzy processing time and fuzzy due date. Multi objective fuzzy job shop scheduling problems are formulated were it aims to maximize the minimum agreement index, average agreement index and minimize the maximum fuzzy completion time. For this purpose, a Genetic algorithm was developed to meet the problem objectives. A new multi objective scatter search (MOSS) approach was proposed by Rahimi-Vahed et al. (2007) for sequencing problem in mixed model assembly line in JIT environment. As the entire objective functions are NP-hard, so obtaining the optimum solution in reasonable amount of time is not possible. Dynamic ideal point was proposed to overcome this problem.

Gokcen and Erel (1997) attempted to develop a binary integer programming model for balancing the mixed-model version of the assembly problems by considering the similarity of precedence relations between different products. The suggested method could avoid rapid increase in the number of decision variables within integer programming model so it would be able to cope with NP-hard nature of mixed-model assembly problems with larger number of variables. Sekar (2007) survey was done on the electronic industry which used printed circuit board (PCB) for mounting electronic components. He aimed to determine the required number of assembly line by principle component analysis (PCA). A binary integer programming model (IP) was proposed in previous research that tried to minimize the make-span of mixed-model assembly lines by optimizing the job allocation on different lines and distributing tasks among workstations using binary integer programming.

This paper presented an approach to address the job allocation problem in high product mix shop floor area. A mixed-integer programming model is presented to formally describe the problem under study and the massive required calculations which make it impossible to be solved optimally. This paper is organized as follows: general descriptions of meta-heuristic methods and the importance of problem under study and the recent achievement in optimization techniques are reviewed in this section. Materials and methods focus on main characteristics of mixed shop floor environment. The proposed mathematical programming model for problem under study is described to interpret the mathematical equations. The complexity of problem under study is also discussed to presents the combinatorial complexity of this problem. In the next step, a proposed hybrid GA-SA mechanism is explained to clarify the implemented hybrid optimization framework. The obtained results of numerical example with both hybrid GA-SA and GA are illustrated in result section and the conclusion and future research of this study is presented in the last section.

MATERIALS AND METHODS

The problem under study includes specific number of mixed-model assembly lines in a shop floor environment where different types of jobs can be processed and each line consist of manual work-stations in which each assembly station is capable of serving task of any jobs. 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). A workstation in any line should be setup for the new materials requirement to be able to serve the new set of products. Each line is equipped with certain number of workstations. Each job consists of a number of tasks that should be done at workstations and has its own initial setup time when it's the first job of the sequence. Change over time is required to change the settings from one job to another in the same line. The following assumptions were considered in this research: All the assembly lines performed assembly operation independently and each assembly line had its own work stations that are capable of serving any type of models. After the job allocation, jobs were not allowed to shift to other assembly lines. Task time was fixed and all tasks were kept at the workstation, once started, until completion. All the times involved in this model such as processing times, change over time, initial setup time and task times are deterministic. Figure 1 show a diagram of problem under study and the engaged parameters within the problem area.

Problem solving procedure

In order to meet the objectives for the considered problem, the proposed solving procedure was composed of two successive stages as follows:

(1) Assigning task to the workstations.
(2) Allocating jobs to the assembly line.

Mathematical model for task assignment

The integer programming model was developed by Sekar (2007) for allocating tasks of a job between multiple workstations so that all the workstations have utmost equal processing time. This model is used for keeping the load balance between workstations which is represented as follows:

$$Min \sum_{i=1}^{N} \sum_{w=1}^{W} D_{iw} \quad (1)$$

Subject to:

$$\sum_{w=1}^{W} X_{iqw} = 1 \; ; \; \{i \in N\}; \{q \in Q\} \quad (2)$$

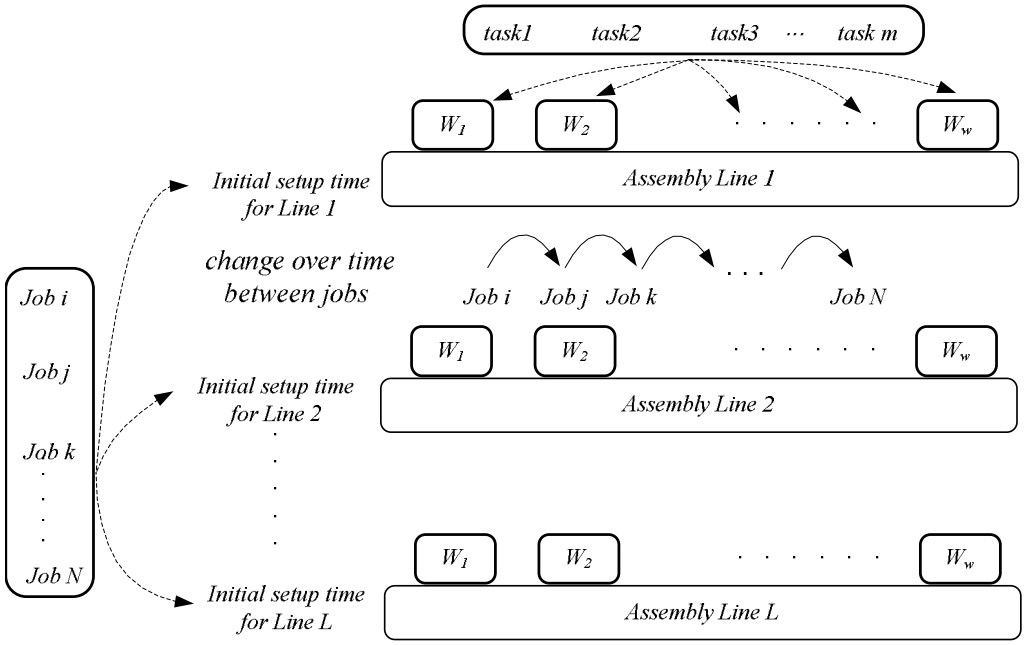

Figure 1. Model diagram of mixed-shop floor and the engaged parameter.

$$\sum_{q=1}^{Q} X_{iqw} \geq 1 \; ; \; \{i \in N\}; \; \{w \in W\} \tag{3}$$

$$\sum_{i=1}^{N}\sum_{q=1}^{Q} X_{iqw} \geq 1; \; \{w \in W\} \tag{4}$$

$$\sum_{q=1}^{Q} t_{iq} \cdot X_{iqw} = L_{iw} \; ; \; \{i \in N\}; \; \{w \in W\} \tag{5}$$

$$Ljs - Liw = Djs \; ; \; \{i \in N\}; \; \{s \in w\}; \tag{6}$$

$$Djs \geq 0; \; \{i \in N\}; \; \{s \in w\}; \tag{7}$$

$X_{iqw} \in 0,1$

X_{iqw} If a task 'q' of a job 'i' assign to 'W'

q is a set of tasks {q∈ Q}

i is the set of jobs {i∈ N}

W is the set of workstations {w ∈ W}

t_{iq} is he processing time for each task 'q' of job 'i'

L_{js} is the processing time of *wth* station for *ith* job

This mathematical model is formulated by the task distribution among workstations for all jobs by minimizing the value of D_{iw} which represents the difference in process time among workstations i, w . This means all workstations have minimum difference in process time for a job. L_{iw} denotes the process time of Wth station for L^{th} job in the system. There are no precedence relations between tasks as they can be served at any workstations. Constraints (2) ensures that each task for a job should be assigned to only one workstation on a line, and once a task is assigned to any workstation, it cannot be reassigned to other workstations. Constraint (3) guarantees that no workstation is left without task assignment and at least one task of a job is assigned to a workstation. Constraint (4) aims to dispense the tasks for all jobs to the workstations without making any limitation for task assignment between jobs. Constraint (5) gives the process time of each workstation for every job by summing all the times of the assigned task to that workstation. D_{iw} represents the difference in process time among work stations for every single job. Constraint (6) aims to compute the value of difference in process time at every work-station for each job. Practically, the value of D_{iw} should be greater than zero. Constraint (7) is used to ensure this objective.

Mathematical programming for job allocation

Each assembly line represents the kind of flow shop system and the workstations representative of involved machines in the flow shop system. Each assembly line acts as flow line system in which the overall make-spam of system is determined by the longest completion of line so minimizing the completion time of all lines directly effect on overall make span of system. The proposed mixed-integer programming model is built based on flow shop model which is developed by Wagner (Pinedo, 2008) 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 follows:

$$Min\{ Max\{ [[Cmax]]_{\perp}\} + \Sigma_{\perp}(e=1)^{\dagger}(L-1)\Sigma_{\perp}(k=e+1)^{\dagger}L B_{\perp}ek + \Sigma_{\perp}(e-1)^{\dagger}(L-1)\Sigma_{\perp}(k-e+1)^{\dagger}L C_{\perp}ek \}$$

which is subject to:

$$Idle_l = \sum_{i=1}^{m-1}\sum_{j=1}^{n}\left(X_{j,1,l} * p_{ij}\right) + \sum_{i=1}^{k_l-1} I_{m,j,l} : l = 1, \dots L \quad (8)$$

$$Setup_l = \sum_{k=1}^{k_l-1}\sum_{i=1}^{n}\sum_{j\neq i=1}^{n}\left(C_{ij} + S_{ik}\right) * \left(X_{i,k_l,l}\right) * \left(X_{j,k_l+1,l}\right) : l = 1, \dots L \quad (9)$$

$$Process_l = \sum_{j=1}^{n}\sum_{k_l=1}^{k_l} X_{j,k_l,l} * P_{mj} \;;\; l = 1, \dots L \quad (10)$$

$$Cmax_l = Setup_l + Idle_l + Process_l \;;\; l = 1, \dots L \quad (11)$$

$$C_{ek} = \sum_{k=e+1}^{L} |Cmax_e - Cmax_k| ; e = 1, \dots .. L-1 \quad (12)$$

$$B_{ek} = \sum_{k=e+1}^{L} |T_e - T_k| \;\; ; \; e = 1, \dots .. L-1 \quad (13)$$

$$T_l = \sum_{j=1}^{n}\sum_{k_l=1}^{k_l} X_{j,k_l,l} * t_j \;:\; L = 1, \dots L \quad (14)$$

$$\sum_{j=1}^{n} X_{j,k_l,l} = 1 \;;\; k_l = 1, \dots k_l \;\; , l = 1, \dots L \quad (15)$$

$$\sum_{k_l=1}^{k_l} X_{j,k_l,l} \leq 1 \;;\; j = 1, \dots n \;\; , l = 1, \dots L \quad (16)$$

$$\sum_{L=1}^{L}\sum_{k_l=1}^{k_l} X_{j,k_l,l} = 1 ; \; j = 1, \dots n \quad (17)$$

$$I_{i,k_l,l} + \sum_{j=1}^{n} X_{j,k_l+1,l} * P_{ij} + W_{i,k_l+1,l} - W_{i,k_l,l} - \sum_{j=1}^{n} X_{j,k_l,l} * P_{i+1,j} - I_{i+1,k_l,l} = 0 \;; \; K = 1, \dots, k_l \;\; 1, i = 1, \dots \dots m \;\; 1, l = 1, \dots L \quad (18)$$

$$Wi1l = 0 \;\; ; i = 1, \dots \dots m-1, l = 1, \dots L \quad (19)$$

$$I_{1k,l} = 0 \;\; ; k = 1, \dots \dots n-1, l = 1, \dots L \quad (20)$$

$$s_{ik} = 0 ; \;\; k = 2 \dots \dots n-1 \quad (21)$$

Where,

$W \geq 0, I \geq 0$
$X_{ijk} \in 0,1$

X_{jkl} If job j is the kth job in the sequence in line L ,
I_{ikl} Ideal time on machine i between the 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 ,
$Idle_l$ Total idle time at the last workstation for lth assembly line,
$Setup_l$ Total setup time for lth assembly line,
$Cmax_l$ Completion time for lth assembly line,
$Process_l$ Processing time at the last workstation of lth assembly line,
C_{ek} Total absolute difference among completion time of assembly line e and the rest of lines,
B_{ek} Total absolute difference among process time of line e and rest of lines, m
t_j Total Process time for job j ,
T_l 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 , and
P_{mj} Process time of job j at workstation m

The first term of objective function 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. The third term of objective is used to minimize the variation of completion time among all assembly lines that has almost finishing time.

Constraint explanation

Minimizing the make span time in *(F_m/Permu/C_{max})* is associated with minimizing the total idle time on the last workstation. The second set of Equation (8) is used to obtain the minimum total idle time at the last workstation for every single assembly line. Equation (9) 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. S_{ik} demonstrates the initial setup time for job i in the k^{th} sequence. It is obvious that initial setup time, only for the first job of sequence has positive value and zero for the rest jobs. S_{ik} *=0; K=2,…,n-1*. There are n jobs in this system and each line processes k_l number of jobs. Equation (10) 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 (11) 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 (12) is used to determine the total absolute difference of completion times for assembly lines. The

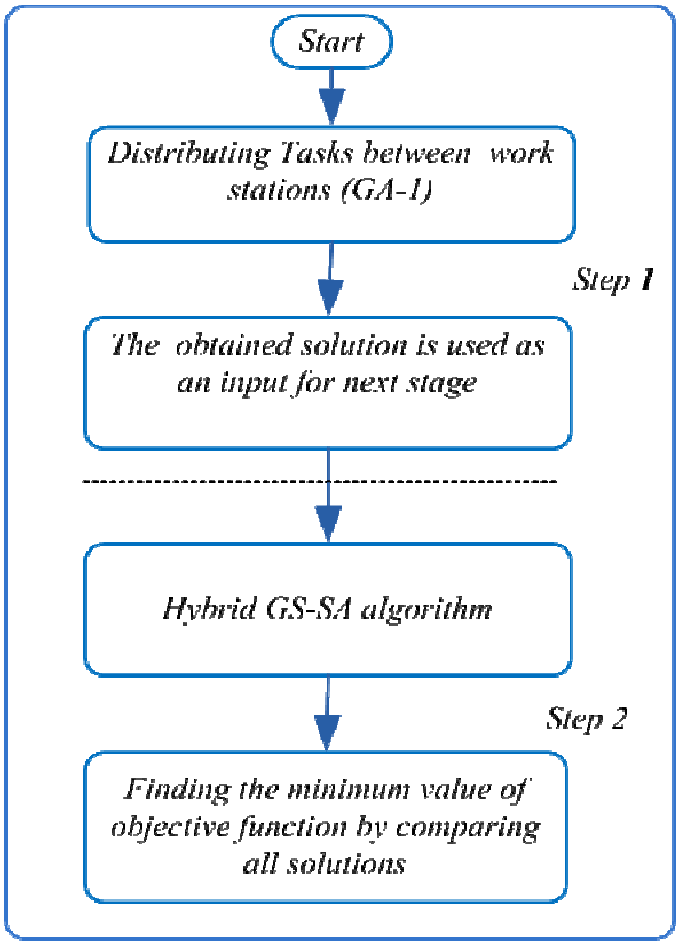

Figure 2. Problem solving procedure by meta-heuristic algorithm.

second term of objective function aims to balance the assembly lines by distributing jobs among multiple lines in which all the lines have utmost equal process time. Equation (13) helps to find the difference in total process time for multiple lines by computing the absolute difference in total process time for lines. Equation (14) is used to attain the process time for every single line which is determined by summing all the jobs allocated to the assembly lines. T_L denotes the total process time of L^{th} assembly.

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 (15) 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 (16) ensures that each job can be placed in only one of the available positions of sequence for each assembly line. Constraint (17) 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 (18) show the inevitable relation between the idle time and waiting time in each assembly line. It represents the logical concept of variables involved in flow shop system. Equation (19) reveals that the waiting time for the first job in a sequence is always equal to zero for any assembly line. Equation (20) shows that the first workstation is always ready to process the first job of a sequence in any assembly line. Equation (21) illustrates that the initial setup time is only considered for the first job of sequence and zero for the rest jobs.

Combinatorial complexity for job allocation problem

In this problem, the total number of permutations for task assignment can be computed as follows:

$$\textbf{Total permutations} = \sum_{c=1}^{Nct} \frac{\left(\sum_{w=1}^{W} k_{wc}\right)!}{\prod_{w=1}^{W}\left(\left(k_{wc}!\right)\right)} \quad (22)$$

W represents the total number of assembly stations and k_{wc} shows the number of tasks assigned to the W^{th} workstation in C^{th} task assignment configuration. N_{ct} represents all the configurations of task assignment to the workstation so the total permutations of all tasks are obtained by summing the task permutations for all configurations. In the job allocation problem, the total number of permutations for job allocation way can be computed as follows:

$$\textbf{Permutations} = \sum_{c=1}^{Nc} N! \quad (23)$$

Where, N_c denotes a set of possible configurations of job allocation to the assembly lines and N represents the number of jobs in the system. It should be noted that the total permutations for this problem is obtained by summing the permutation for all possible configurations of job allocation. So, in order to solve the problems by mathematical methods, different values of input parameters k_l for different configurations of job allocation should be formulated. In this case, each set of equations only represents a specific configuration of job allocation. In order to check all the configurations of job allocation, the proposed model should be formulated for several times and it requires huge amount of mathematical calculations and modeling which is quite time consuming and inefficient. Thus, problem with larger number of variables usually cannot be solved within a reasonable amount of time. The main objective of this research is to provide an evolutionary-based solving procedure to overcome the mathematical techniques limitations and find the solutions in an efficient way.

Meta-heuristic algorithm

The solving procedure starts by finding the best task allocation in which it balances the workstations. Allocating tasks to the workstations which minimizes the total time difference between work stations is the purpose of the GA-1. It gives the workload for every single job at each workstation which is required to compute the make-span for each line. In next stage, hybrid GA-SA tries to allocate jobs to the assembly lines in order to minimize the multi objective functions for the considered problem. Assigning jobs to the assembly lines can be done in different configuration of job allocation. So, in order to find the best solution for this problem, hybrid GA-SA should run for all possible job allocation configurations. The entire solving procedure by meta-heuristic methods is shown in Figure 2.

Genetic algorithm for task assignment problem (GA-1)

Fitness function is used to evaluate the generated chromosomes to measure the optimality of solutions. The chromosome is a string of length Q (number of tasks for job j) that k_{wc} corresponds to the

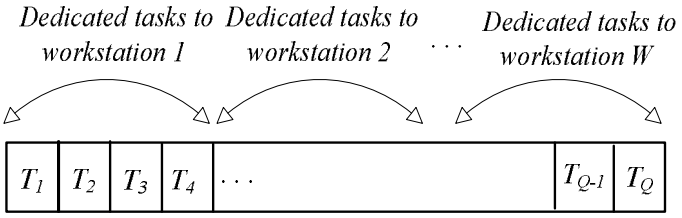

Figure 3. Chromosome of task for a job.

number of tasks assign to w^{th} station in the C^{th} configuration. Figure 3 show a chromosome of tasks and also show how they are assigned to the workstations. A fitness function for task assignment problem is shown in Equation (24) where the main objective focuses on minimizing the absolute value of total differences in processing time among all possible workstations. The best value of F_w can be achieved while all stations are fully balanced so, in that case the objective is 100% met and while the absolute value of total difference in processing times increases, F_w tends to zero. In order to guaranty the best task allocation, all possible configurations of task allocation is required to be checked and minimum value of unbalanced time is selected as best tasks allocation configuration for that particular job. Controlling the balancing parameter for this problem is computed by the proposed fitness function which is given by

$$F_w = 1/\left(1 + \sum_{s=1}^{W-1}\sum_{k=s+1}^{W}|T_{js} - T_{jk}|\right) \; ; \quad j = 1,2,\dots,N \tag{24}$$

Where T_{jw} is the total process time for a job j on the W^{th} workstation given by

$$T_{j1} = \sum_{q=1}^{k_{1c}} t_{jq} \;, T_{j2} = \sum_{q=k_{1c}+1}^{k_{1c}+k_{2c}} t_{jq} \;, \dots, T_{jw} = \sum_{q=\sum_{i=1}^{W-1} k_{wc}+1}^{Q} t_{jq} \; ; \; j = 1,\dots,N; k_{wc} \in Nc \tag{25}$$

Where,

t_{jq} is the processing time of qth task in sequence of job j ,

Q is the total number of tasks required to complete a job,

Nc is a set of possible configurations for task assignment to workstations, and

k_{wc} is the number of jobs assigned to the wth workstation in Cth configuration.

Hybrid GA-SA algorithm for job allocation problem

Genetic algorithm and simulated annealing are categorized as global search heuristics techniques which are able to tackle the complexity of large size problems by finding near optimal solutions. SA evolution mechanism perform based on iterative process of modifying and examining in the neighborhood of current solution (local search) while GA handles set of potential solutions which are based on retaining and transferring the useful information during the generations (Wang and Zheng, 2001). The hybrid approach combines both advantages of GA for global searches and the advantage of SA for local ones by using the respective advantages (Oysu and Bingul, 2009). An effective combination of GA and SA, hybrid GA-SA is developed to solve the scheduling problem.

A chromosome of job is generated in a predetermined number of population and the cost evaluation for each chromosome is done in the next step. A proportion of the existing population is selected through a fitness-based process in each generation which is more likely to produce better solutions for the next generation. The next step is to generate a second generation of solutions from those selected through genetic operators. For this means, some chromosomes are randomly selected for reproduction operation. In order to distract the algorithm from reaching the premature convergence, mutation is randomly applied on few chromosomes. The best individual in each generation is selected for neighborhood search by SA. The neighborhood search is executed within the job chromosomes to increase the quality of obtained solution. When the simulated algorithm finishes, the enhanced chromosome is copied to the population for further evolution through exploring other potential solutions. This process will be continued until the stopping criteria are met. The proposed hybrid GA-SA is shown in Figure 4.

Generally, the appropriate fitness function closely associates with mathematical objective function which is capable of computing the cost for each chromosome quickly. As can be seen from Figure 5, the chromosome is a string of length N where K_{lc}; $l=1,2,\dots,L$ represents the number of jobs assigns to the L^{th} assembly line in the C^{th} configuration so for finding optimum solution to this problem, different configuration of fitness function computing should be done. 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

$$F_{MBC} = 1/(M + B + C)^2 \tag{26}$$

Where,

$$M = Max\{[[Cmax]]_{\downarrow i}\} \tag{27}$$

$$B = \sum_{s=1}^{L-1}\sum_{k=s+1}^{L}|T_s - T_k| \tag{28}$$

$$C = \sum_{s=1}^{L-1}\sum_{k=s+1}^{L}|Cmax_s - Cmax_k \tag{29}$$

Initial population

The population in both GA-1 and hybrid GA-SA are fixed during all generations and it is determined base on complexity of the problems to provide required diversity for initial population. The population size for GA-1 is set to 30 in each generation. The complexity of job allocation problem is increasing by $N!$ order. So a larger population is required to provide more diversity of potential solutions and discourages premature convergence to local optimums. The population size for hybrid algorithm is set to 80 which are fixed in each generation. Total number of generation is used as a stopping criterion for both GA-1 and hybrid GA-SA programs. GA-1 is terminated when it reaches 100 generations and the genetic part of hybrid heuristic requires more iteration to obtain

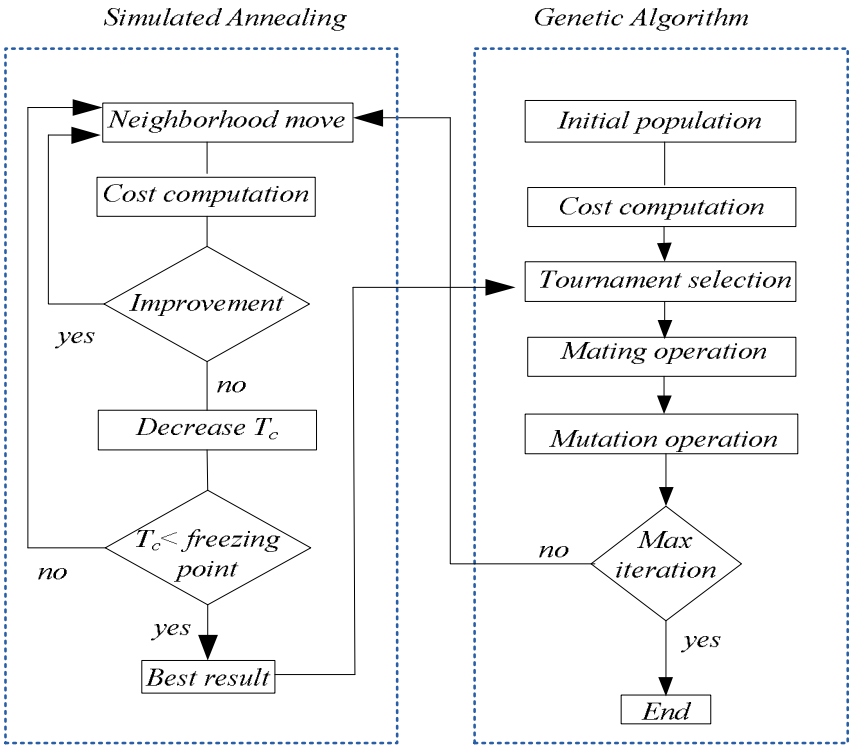

Figure 4. Hybrid GA-SA algorithm.

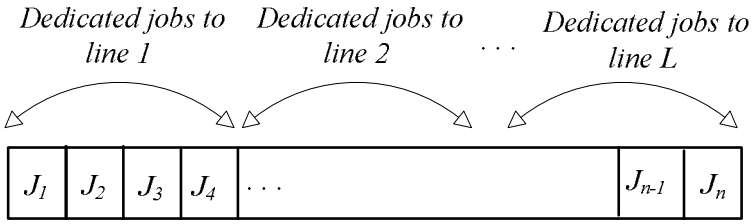

Figure 5. A Chromosome of jobs.

the potential solutions, so the algorithm terminates at 300 generations.

Tournament selection

Tournament selection is a very popular strategy that aims to imitate natural competition of specious (Michalewicz, 1996). The tournament selection works in a 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 (Eiben and Smith, 2003).

Crossover

Crossover is a genetic operator that combines sets of information from different chromosomes and generates new offspring to capture both individual's information (Ariffin et al., 2004). It is probable that superior parents may produce better offspring. Partially Mapped Crossover (PMX) is employed as crossover operator in both GA-1 and hybrid GA-SA algorithms while GA-1 attempts to find the best task distribution whereas hybrid GA-SA aims to find the best sequence of jobs within each line. The crossover rate is set based on the initial population size for each program in order to increase the algorithm efficiency. Meanwhile, a population size of 30 with an 80% crossover rate is applied for GA-1. For the hybrid GA-SA algorithm, a 50% crossover rate is considered appropriate, which is able to find a good solution in a reasonable amount of time (Grefenstette, 1986). Meanwhile, population size of 30 with 80% crossover rate is applied for (GA-1). For hybrid GA-SA algorithm, 50% crossover rate is considered appropriate which is able to find good solution in a reasonable amount of time (Grefenstette, 1986).

Mutation

Mutation operator aims to provide a means to prevent algorithm

Table 1. All configurations of task assignment to the two workstations for seven tasks.

Configuration	Number of tasks assigned to W_1	Number of tasks assigned to W_2
	K_{1c}	$K_{2c} = Q - K_{1c}$
1	4	3
2	5	2
3	6	1

Table 2. Different configurations of Job allocation to the lines.

Number of jobs assigned to Line 1	Number of jobs assigned to Line 2	Number of jobs assigned to Line 3
K_{1c}	K_{2c}	$K_{3c} = N - K_{1c} - K_{2c}$

from rapid convergence or premature convergence and drive algorithm to search for further feasible problem space to escape from local optimum, this means swap mutation is elected as mutation operator. The Mutation probability is set to 0.02 in both GA-1 and genetic programming of hybrid algorithms which is a typical value for Genetic algorithm (Leu et al., 1994).

Neighborhood search

In hybrid algorithm, the best obtained solution in each genetic algorithm generation is transferred to SA in order to improve the quality of solution through neighborhood search to produce a solution close to the current solution in search space, by randomly swapping the positions of two elements in a chromosome. Thus, a new solution is produced (Leung et al., 2001).

Cooling scheduling

In simulated annealing algorithm, worse moves might be accepted based on current temperature to avoid algorithm to stick in a local optimum and the acceptance probability for worse moves, decreasing as the algorithm proceeds to the freezing point. The exponential cooling scheduling is shown as an appropriate performance in simulated annealing algorithm as it's able to compromise between fast schedule and also the ability to reach lower energy state (Wang and Zheng, 2001). The exponential cooling schedule is given by $T_k = \alpha * T_{k-1}$ where $\alpha \epsilon (0,1)$ is the temperature decrease rate. The initial experiment was demonstrated to us that the initial temperature of 30 is suitable to explore the potential solutions by neighborhood movement. Freezing point is set at 1 and the temperature is decreasing by factor of 0.05 $\alpha = 0.05$.

Elitism

Elitism is usually used to prevent the loss of the current fittest member of the population due to crossover or mutation operators or neighborhood search and keep the best solution through the entire hybrid search process (Haupt and Haupt, 1998).

RESULTS AND DISCUSSION

In this paper a numerical example is used to verify the efficiency of a proposed hybrid algorithm by comparing the obtained result by a simple genetic algorithm. The experiment is carried out on one set of problem tests (Sekar, 2007). The first step of the proposed procedure is to compute the best task assignment for every single job. Hybrid GA-SA attempts to determine the best job allocation configuration and the optimum sequence of assigned jobs to each line that meet all the considered objectives in the problem. The total number of the jobs in the system is thirteen with maximum seven tasks per job. Three assembly lines are selected to process thirteen jobs and each line equipped with two workstations. Several configurations of task assignment and job allocation in both problems are available. The GA-1 starts from first job and attempts to assign tasks to the workstations. All possible configurations for task assignment are displayed in Tables 1 and 2. For example the first configuration illustrates that four tasks are assigned to the first workstation and the rest of three are assigned to the second workstation.

Example: mK1=5,K2=5,K3=3 represents that 5, 5 and 3 jobs are allocated to the line1, 2 and 3 respectively. By this means fourteen different configurations of job allocation are available for thirteen jobs and three lines so each configuration is considered as a new problem which is solved by hybrid GA-SA.

Input data in numerical example

The hybrid GA-SA algorithm and simulated annealing are coded in MATLAB 7.1 and run on a 1.66 GHz core2 CPU computer. The data used in this research are represents in the following tables. Task time for every single job is shown in Table 3. Table 4 includes the initial setup time and change over time matrix for all involved jobs.

Table 5 illustrates the task assignment to workstations that is obtained at the end of the first stage. The first column represents the number of jobs in the system. The second and third columns show the workload of each workstation for each job. The workload difference between two workstations for each job is shown in the

Table 3. GA-1, Input: Task processing times.

Jobs\Tasks	1	2	3	4	5	6	7	Process time
job 1	12.3	35.53	34.85	4.78	6.15	19.13	16.4	129
job 2	22.95	11.05	0	5.95	7.65	11.9	13.6	73
job 3	15.15	65.65	28.62	11.78	30.3	70.7	26.93	249
job 4	12.6	9.1	17.85	4.9	3.15	4.9	8.4	61
job 5	22.8	82.33	43.07	35.47	22.8	0	40.53	247
job 6	56.7	27.3	35.7	44.1	18.9	58.8	33.6	275
job 7	37.8	81.9	35.7	29.4	18.9	58.8	0	263
job 8	30.6	22.1	28.9	17.85	7.65	0	27.2	134
job 9	11.25	27.08	14.17	11.67	7.5	5.83	10	88
job 10	58.95	56.77	111.35	15.28	39.3	61.13	34.93	378
job 11	34.2	49.4	21.53	35.47	22.8	35.47	10.13	209
job 12	27.45	39.65	34.57	21.35	18.3	42.7	24.4	208
job 13	36.45	35.1	45.9	28.35	12.15	37.8	10.8	207

Table 4. GA-SA Input: Initial setup time and change over time of Jobs.

Job/Job	j 1	j 2	J 3	j4	j5	j 6	j 7	j 8	j 9	j 10	j 11	j 12	j13	Initial setup time
j 1	0	10	9	8	10	11	12	11	9	7	13	5	14	25
j 2	10	0	12	16	17	8	6	15	13	7	10	9	16	30
j 3	9	12	0	19	7	16	12	14	13	18	19	20	12	32
j 4	8	16	19	0	13	18	11	8	19	16	11	7	5	22
j 5	10	17	7	13	0	17	15	20	12	19	13	16	8	35
j 6	11	8	16	18	17	0	9	10	8	6	10	11	17	33
j 7	12	6	12	11	15	9	0	6	15	13	12	10	19	35
j 8	11	15	14	8	20	10	6	0	5	16	11	18	10	39
j 9	9	13	13	19	12	8	15	5	0	14	14	5	7	33
j 10	7	7	18	16	19	6	13	16	14	0	11	13	9	29
j 11	13	10	19	11	13	10	12	11	14	11	0	6	14	37
j12	5	9	20	7	16	11	10	18	5	13	6	0	6	28
j 13	14	16	12	5	8	17	19	10	7	9	14	6	0	36

Table 5. GA-1 output: Workload for every single job at workstations.

Job	Workload at workstation 1	Workload at workstation 2	Unbalanced time	Task distribution	
				Workstation 1	Workstation 2
1	64.91	64.23	0.68	6 5 3 4	1 7 2
2	36.55	36.55	0	6 4 5 2	1 3 7
3	124.56	124.57	0.01	1 6 4 7	2 3 5
4	30.45	30.47	0.02	7 4 6 5	2 3 1
5	124.14	122.86	1.28	1 5 4 3	6 7 2
6	136.5	138.6	2.1	1 7 5 2	6 3 4
7	132.3	130.2	2.1	6 7 1 3	5 2 4
8	67.15	67.15	0	5 3 1 6	4 7 2
9	43.34	44.16	0.82	7 3 4 5	1 6 2
10	189.95	187.76	2.19	5 2 7 1	3 4 6
11	103.87	105.13	1.26	5 7 6 4	3 1 2
12	103.7	104.72	1.02	5 2 4 7	1 6 3
13	103.95	102.6	1.35	5 7 3 2	6 4 1

Table 6. Job allocation results for all configurations of job allocation.

Configuration of job allocation	Make-span		Process time difference		Completion time difference		Total objective		Objective difference
	Hybrid	GA	Hybrid	GA	Hybrid	GA	Hybrid	GA	[Hybrid – GA]
K_1 = 11, k_2 =1, k_3 =	1153.32	1153.32	3186	3186	1690.44	1690.44	6029.76	6029.76	0
K_1 = 10, k_2 =2, k_3 =	1015.8	1015.8	2454	2454	1218.18	1218.18	4687.98	4687.98	0
K_1 = 9, k_2 =2, k_3 =	882.92	883.52	1636	1636	863.04	864.24	3381.96	3383.8	1.84
K_1 = 9, k_2 =3, k_3 =	884.92	884.34	1956	1956	956.42	955.26	3797.34	3795.6	-1.74
K_1 = 8, k_2 =3, k_3 =	730.74	730.39	912	912	355.06	354.36	1997.8	1996.7	-1.1
K_1 = 8, k_2 =4, k_3 =	730.39	733.04	1462	1462	647.36	652.66	2839.75	2847.7	7.95
K_1 = 7, k_2 =3, k_3 =	649.17	649.17	230	230	108.4	108.4	987.57	987.57	0
K_1 = 7, k_2 =4, k_3 =	663.02	666.82	572	572	219.62	227.22	1454.64	1466	11.36
K_1 = 7, k_2 =5, k_3 =	734.15	664.8	1390	1390	654.88	654.88	2779.03	2779.03	0
K_1 = 6, k_2 =5, k_3 =	649.17	649.17	44	36	4.5	27.48	697.67	712.65	14.98
K_1 = 6, k_2 =5, k_3 =	656.41	655.97	576	588	206.4	205.52	1438.81	1449.5	10.69
K_1 = 6, k_2 =6, k_3 =	734.81	740.06	1390	1388	656.2	666.7	2781.01	27948.	13.79
K_1 = 5, k_2 =4, k_3 =	654.37	654.37	10	10	30.48	30.48	694.85	694.85	0
K_1 = 5, k_2 =5, k_3 =	656.48	656.48	24	24	6.46	6.46	686.94	686.94	0

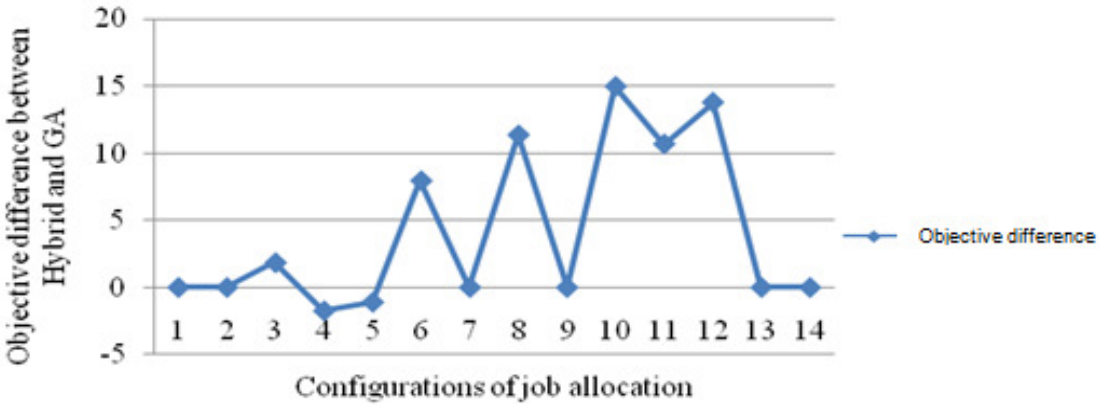

Figure 6. Total objective difference between hybrid algorithm and Genetic algorithm.

fourth column which is inevitable for some jobs. Finally the fifth column illustrates the task distribution between workstations. The process time for every single job at both workstations obtained by GA-1 is used as an input for hybrid GA-SA for computing the final completion time of assembly lines. The obtained results for all configurations of job allocation with both hybrid GA-SA and Genetic Algorithm for all the individual sub-objectives are shown in Table 6. Figure 6 represents a total objective comparison between hybrid GA-SA and GA, which reveals that the hybrid algorithm provides superior performance to GA as it reaches better solutions in most of the configurations of job allocation. By comparing the total objective values between all configurations of job allocation, the minimum value of total objective function is achieved in k1=5, k2=5, and k3=3 as both hybrid GA-SA and GA reach the same solution in all objectives. Table 7 reveals the job sequence in each line that provides an optimum solution in which all lines have almost the same make-span and completion times. As can be seen from Table 7, the longest completion time determines the overall make-span of the system. The maximum difference between all completion times is only 3.23 time units (t.u), which clearly shows that all lines have almost the same completion time. The maximum difference between total process times in each line is 12 t.u, which reveals that all lines are closely balanced. A comparison between all configurations of job allocation for all objectives is shown in Figure 7. The comparison of obtained results revealed that the hybrid GA-SA algorithm produced relatively better results than the GA.

Conclusion

In this paper, a hybrid GA-SA procedure is applied to

Table 7. GA-SA Output: Best Job sequence is obtained by $K_1 = 5$, $k_2 = 5$, $k_3 = 3$.

Line	Line 1					Line 2					Line 3			M	B	C	Objective
Completion time	656.48					653.25					654.88			656.48	6.46		
Process time	841					846					834					24	686.94
Job sequence	1	9	4	6	12	8	5	2	1	7	10	13	3				

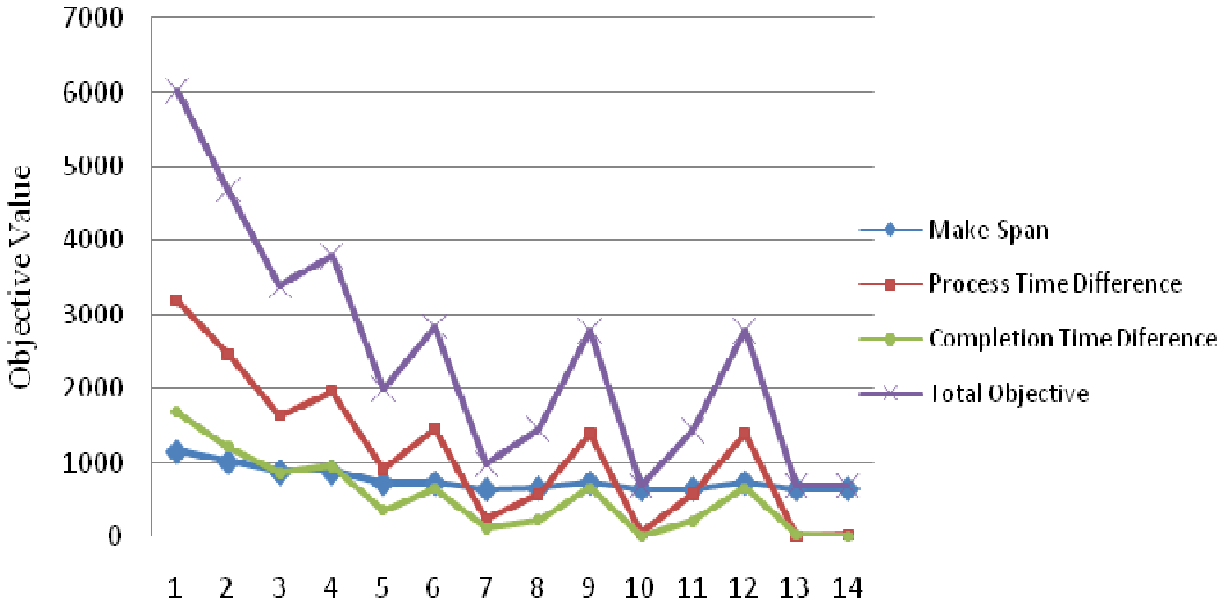

Figure 7. Total objective comparison for all configurations of job allocation.

tackle the complexity of sequencing problems in parallel mixed-model assembly lines. For solving such problems by mathematical methods, the proposed mixed-integer model should be formulated for several configurations of job allocations, which is quite time consuming and inefficient. A hybrid GA-SA is implemented to overcome the massive search space needed for optimizing the multi-objective function. The solving procedure starts by finding the best task allocation based on the processing time of the tasks that balance the workstations by simple GA-1. Allocating tasks to the workstations to minimize the total time difference between workstations is the purpose of the GA-1. In the next stage, hybrid GA-SA attempts to allocate jobs to the assembly lines in order to minimize the multi-objective functions. In order to check the efficiency of the proposed search algorithm, a genetic algorithm is also applied and the obtained results from both meta-heuristic methods are compared. The comparisons with the single GA and hybrid GA-SA approaches prove that the hybrid approach can result in more satisfactory results.

However, future work should implement a systematic solving procedure for a larger number of configurations of job allocation. In such problems with additional input parameters, more configurations of job allocation would be available so that a systematic and intelligent solving procedure could be developed to tackle the problem with higher degrees of complexity.

REFERENCES

Eiben AE, Smith JE (2003). Introduction to evolutionary computing. springer.

Gokcen H, Erel E (1997). A goal programming approach to mixed-model assembly line balancing problem. Int. J. Product. Econ.. 48(2): 177-185.

Grefenstette JJ (1986). Optimization of control parameters for genetic algorithms. IEEE Transactions on Systems, Man and Cybernetics, 16(1): 122-128.

Groover MP (2001). Automation, production systems, and computer-integrated manufacturing. London : Prentice Hall ; Prentice-Hall International, Upper Saddle River, N.J., xv, p. 856.

Haupt RL, Haupt SE (1998). Practical genetic algorithms. Wiley New York.

Kim YK, Hyun CJ, Kim Y (1996). Sequencing in mixed model assembly lines: A genetic algorithm approach. Comput. Operat. Res., 23(12): 1131-1145.

Leu YY, Matheson LA, Rees LP (1994). Assembly Line Balancing Using Genetic Algorithms with Heuristic-Generated Initial Populations and Multiple Evaluation Criteria*. Decision Sci., 25(4): 581-606.

Leung TW, Yung CH, Troutt MD (2001). Applications of genetic search and simulated annealing to the two-dimensional non-guillotine cutting stock problem. Comput. Ind. Eng., 40(3): 201-214.

Michalewicz Z (1996). Genetic algorithms+ data structures= evolution

programs. Springer.
Mohd Ariffin MKA, Sims ND, Worden K (2004). Genetic optimization of machine tool paths. Adaptive computing in design and manufacturing VI, pp. 125-135.
Oysu C, Bingul Z (2009). Application of heuristic and hybrid-GASA algorithms to tool-path optimization problem for minimizing airtime during machining. Eng. Appl. Artif. Intel., 22(3): 389-396.
Pinedo ML (2008). Scheduling: theory, algorithms and systems. Springer.
Rahimi-Vahed AR, Rabbani M, Tavakkoli-Moghaddam R, Torabi SA, Jolai F (2007). A multi-objective scatter search for a mixed-model assembly line sequencing problem. Adv. Eng. Inf., 21(1): 85-99.
Rehg JA (1994). Computer-integrated manufacturing. Prentice Hall.
Sakawa M, Kubota R (2000). Fuzzy programming for multiobjective job shop scheduling with fuzzy processing time and fuzzy duedate through genetic algorithms. Europ. J. Operat. Res., 120(2): 393-407.
Sekar V (2007). Minimizing the make-span in a high-product mix shop-floor using integer programming. M.S. Thesis, State University of New York at Binghamton, United States -- New York.
Simaria AS, Vilarinho PM (2009). 2-ANTBAL: An ant colony optimisation algorithm for balancing two-sided assembly lines. Comput. Industr. Eng., 56(2): 489-506.
Stevenson William J (2005). Operation Management. McGraw-Hill international edition, p. 38.
Sule DR (1997). Industrial scheduling. PWS Pub. Co Boston.
Wang L, Zheng DZ (2001). An effective hybrid optimization strategy for job-shop scheduling problems. Comput. Operat. Res.,28(6): 585-596.
Zandieh M, Fatemi Ghomi SMT, Moattar Husseini SM (2006). An immune algorithm approach to hybrid flow shops scheduling with sequence-dependent setup times. Appl. Math. Comput., 180(1): 111-127.