RESEARCH ARTICLE

ScienceAsia 35 (2009): **295–305**

doi: 10.2306/scienceasia1513-1874.2009.35.295

Multi-objective sequencing problems of mixed-model assembly systems using memetic algorithms

Parames Chutima*, **Penpak Pinkoompee**

Department of Industrial Engineering, Faculty of Engineering, Chulalongkorn University, Phayathai Road, Bangkok 10330, Thailand

*Corresponding author, e-mail: Parames.C@chula.ac.th

Received 2 Feb 2009
Accepted 30 Jun 2009

ABSTRACT: This paper investigates the performance of local searches embedded in memetic algorithms for solving multi-objective mixed-model assembly line sequencing problems that are common in a just-in-time production system. Two inversely related objectives, namely, setup times and production rate variation, are simultaneously considered. We use memetic algorithms which are a type of evolutionary algorithm using a local search algorithm to exercise exploitation. Simulation results demonstrate that memetic algorithms employed in conjunction with an appropriate local search outperform highly meta-heuristic algorithms such as Strength Pareto Evolutionary Algorithm 2 and Non-dominated Sorting Genetic Algorithm II in terms of ability to find Pareto-optimal solutions.

KEYWORDS: local search algorithms

INTRODUCTION

Mass production of a single standardized product model has helped manufacturers in many industries such as automobiles, appliances, and toys to save costs substantially for several decades. However, intensive competition in the current market and ever-changing customer requirements have meant that producing a large volume of a single product on a single assembly line is no longer cost effective. In order to offer more varieties of products to attract their customers, manufacturers need to redesign their production lines to accommodate mixed-model production which are known as mixed model assembly lines (MMALs).

MMAL is a production line where a diversified small-lot product models with similar characteristics (e.g., shape, size, colour, process requirements) are assembled. If an appropriate mixing sequence of product models is launched into the MMAL, the line will be operated efficiently without keeping large inventories. MMALs are widely employed in just-in-time (JIT) production systems. Their flexibilities provide more opportunities for a manufacturing firm to be responsive to changes in the external environment. To use such MMALs in an efficient manner, line balancing and model sequencing are necessary. Line balancing is the problem of determining the cycle time and number and sequence of workstations on the line to accommodate the different models of products. Model sequencing is a problem of determining a production sequence of mixed models introduced to the line to achieve given objectives. These two problems are normally solved sequentially. In this paper, we assume that the line has already been balanced and only the sequencing problem is considered.

Determining the intermixed sequence of introducing models to the MMAL is vital to the efficient use of a JIT production concept. Two basic goals for the sequencing problem were originated by Monden[1]: (1) smoothing the workload (total operation times at each workstation on the assembly line) and (2) maintaining a constant rate of usage for every part consumed by the assembly line. Toyota Corporation developed Goal Chasing I (GC-I) and II (GC-II) methods to solve these problems. GC-I selects an order of the models to launch in the assembly line so that the one-stage variation at each stage is minimized, and GC-II simplifies GC-I and is solved under special assumptions regarding product structure. Here a ‘stage’ represents a position in the model launch order of a sequence.

Miltenburg[2] developed non-linear integer programming to minimize the total deviation of actual production rates from the desired production rates. Since the time complexity function of the proposed program was exponential, two heuristics were proposed to solve the problem. Miltenburg et al[3] proposed a dynamic programming algorithm to solve the same problem. However, the algorithm seems to be inapplicable to large-sized problems. Kubiak and

Sethi[4] developed an optimization algorithm to solve GC-II of Monden[1]. They showed that the objective function can be represented by using penalty functions for the deviation from most even. Moreover, the problem can be reduced to an assignment problem if these penalty functions are non-negative and convex. Korkmazel and Meral[5] showed that the bi-criteria problem of Monden[1] with the sum-of-deviations type objective can be formulated and solved as an assignment problem.

Recently, the sequencing problem of MMALs has been focused on multi-objective criteria. Bard et al[6] proposed Tabu Search (TS) approach with weighted sum objectives to minimize the overall line length while keeping a constant rate of part usage. The Pareto Stratum-Niche Cubicle Genetic Algorithm (PS-NC GA) for finding the best sequence satisfying the three objectives of minimizing total utility work, keeping a constant rate of part usage, and minimizing total setup cost was proposed by Hyun et al[7]. McMullen[8] developed the TS algorithm for the problem which considered the two objectives of minimizing the part usage rate as defined by Miltenburg[2] and minimizing the number of setups. McMullen and Frazier[9] developed a Simulated Annealing (SA) algorithm and compared it with TS. The results showed that SA was superior to TS. McMullen[10] developed a Genetic Algorithm (GA) and compared it with TS and SA. They found that the performance of TS was no better than SA and GA. Furthermore, McMullen[11] solved the same problem by using GA and Kohonen's self-organizing map (SOM). Ant Colony Optimization (ACO) was also proposed by McMullen[12] and was compared against SA, TS, GA, and SOM. The results showed that ACO had the best performance. Mansouri[13] proposed Multi-Objective Genetic Algorithm (MOGA) to solve the problem proposed by McMullen[10]. Tavakkoli-Moghaddam and Rahim-Vahed[14] proposed Memetic algorithm (MA) with a weighted sum objective and solved the problem proposed by Hyun et al[7]. MA was reported to find promising results, especially for large-sized problems. Rahim-Vahed et al[15] proposed a new multi-objective scatter search (MOSS) to solve the problem proposed by Hyun et al[7] and compared it with PS-NC GA, non-dominated sorting genetic algorithm II (NSGA II), and strength Pareto evolutionary algorithm 2 (SPEA 2). The computational results showed that the proposed MOSS outperformed all of these GAs. However, it took much longer to find solutions, especially on large-scale problems.

Most prior research has been directed towards the development of efficient approximation algorithms or heuristics to find the best mixed-model sequence. Memetic algorithms (MAs) seem to be outstanding candidates for this. However, the MAs that have been developed so far have mostly focused on single-objective or weighted sum objective optimization problems[16]. Although the newer heuristics such as MOSS performs well on multi-objectives, they take longer to find Pareto optimal solutions than basic heuristics such as SPEA 2 and NSGA II. As a result, in this research we propose a new MA that combines modified SPEA 2 and NSGA II (which are good at finding Pareto optimal solutions quickly) with some powerful local searches (which are good at improving solution quality) and call them Memetic-SPEA 2 (MSPEA 2) and Memetic-NSGA II (MNSGA II).

The objective of this paper is to study the results of MA for solving multi-objective sequencing on MMAL in a JIT production system. The primary research questions are the followings: (1) Will the solution quality of MSPEA 2 and MNSGA II be better than SPEA 2 and NSGA II? (2) If so, which one is better? (3) What local search applied in MSPEA 2 and MNSGA II gives the better performance? The problem sets to be tested in this research were those proposed by McMullen[10] with two objectives to be simultaneously optimized: (1) minimizing setup times and (2) minimizing variation of production rates. Although most literature assumes that setup times between different products are negligible, in reality there are some applications such as in electronics industry where setup times may represent a high proportion of the cycle time and also be dependent on the directed preceding model. Hence, to make the model of the MMAL more realistic, sequence-dependent setup times are considered in this research.

MULTI-OBJECTIVE EVOLUTIONARY ALGORITHMS

A multi-objective optimization problem (MOP) is related to the problem where two or more objectives have to be optimized concurrently. Generally, such objectives are conflicting and are represented in different measurement units, preventing simultaneous optimizations of each one. MOP can be formulated, without loss of generality, as follows:

$$\underset{x\in\Omega}{\text{Minimize}}\, f_1(x), f_2(x), \ldots, f_k(x), \tag{1}$$

where solution x is a vector of decision variable for the considered problem, Ω is the feasible solution space and $f_i(\cdot)$ is the ith objective function ($i = 1$, 2, …, k). Two approaches can be employed to solve MOP. The first approach is to combine each objective function into a single composite function using e.g., the weighted sum method or utility theory. Two practical problems are often experienced with this approach: (1) selection of the suitable weights can be very difficult even for those who are familiar with the problem and (2) small perturbations in the weights can sometimes lead to totally different solutions[17]. The second approach, using multi-objective evolutionary algorithms (MOEAs), is to determine a set of alternative solutions for (1) rather than a single optimal solution. These solutions are optimal in the wider sense that no other solutions in the search space are superior to them when all objectives are considered. A decision vector x is said to dominate a decision vector y (also written as $x \succ y$) if:

$$f_i(x) \leqslant f_i(y) \text{ for all } i \in \{1, 2, \dots, k\}, \quad (2)$$

$$f_i(x) < f_i(y) \text{ for at least one } i \in \{1, 2, \dots, k\}. \quad (3)$$

All decision vectors that are not dominated by any other decision vector are called *non-dominated solutions* or *Pareto-optimal solutions*. If non-dominated solutions are abundant, located close to each other, and distributed uniformly, the contour of these non-dominated solutions will appear as a curve. This curve is called the *Pareto front* or *non-dominated front*. Normally, the *true-Pareto front* that comprises all global-optimal non-dominated solutions is unknown. Hence, the best Pareto front found after an extensive search for non-dominated solutions is completed is used as a representative for the true-Pareto front and is called the *approximated true Pareto front*. The Pareto-optimal solutions for a two-objective minimization problem are illustrated in Fig. 1. It is obvious that an amount of sacrifice in one objective is always incurred to achieve a certain amount of gain in the other while moving from one Pareto-optimal solution to another. Providing Pareto-optimal solutions to the decision maker is more preferable to a single solution since practically, when considering real-life problems, a final decision is always based on a trade-off between conflicting objective functions.

MOEAs have recently become popular and have been applied to a wide rage of problems from social to engineering problems[18]. In general, MOEAs are ideally suited to MOP because they are capable of searching multiple Pareto-optimal solutions in a single run. The approximation of true Pareto-optimal set involves two conflicting objectives: (1) the distance to the true Pareto front is to be minimized, and (2) the diversity of the evolved solutions is to be maximized[19]. To achieve the first objective, a Pareto-based fitness assignment is normally designed to guide the search towards the true Pareto-optimal front[20, 21]. In the view of the second objective, some MOEAs successfully provide density estimation methods to preserve the population diversity. Although several versions of MOEAs have been developed[18], SPEA 2[22] and NSGA II[23] are among the most promising ones in terms of convergence speed and good Pareto-optimal solutions and even distribution of the Pareto front.

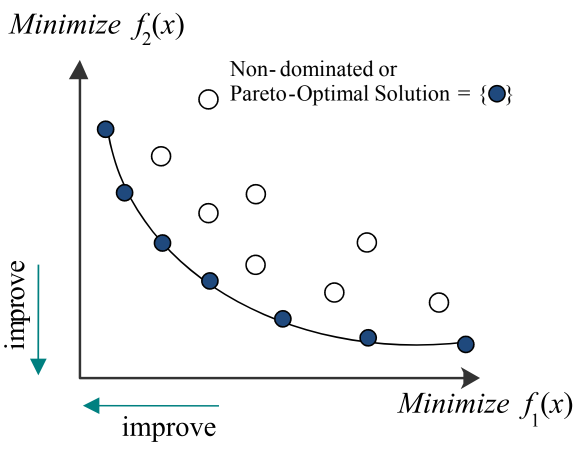

Fig. 1 Pareto-optimal solutions.

SPEA 2[22] is an improved version of SPEA[24] for multi-objective optimization problems. The improvements of SPEA 2 over SPEA include a fine-grained fitness assignment scheme, a density estimation technique based on the k-nearest neighbour metric, and an archive truncation method for preserving boundary solutions. In SPEA 2, the size of the archive (a place where non-dominated solutions found so far are kept) is fixed. If the number of non-dominated individuals exceeds the archive size, the truncation operator is activated by evaluating the k-nearest neighbour metric and the members that are in the most crowded areas of the front are removed. In contrast, if the number of non-dominated individuals is less than the archive size, the remaining archive is filled with dominated individuals that win the binary tournament selection.

NSGA II is similar to SPEA 2 since they both employ an elitism algorithm, but they differ in the elitism preservation operation. Elitism of NSGA II does not use secondary external populations or archive populations, whereas SPEA 2 uses archive external list to store non-dominated solutions discovered so far in the search. NSGA II sets the archive size equal to the initial population size. The current archive is determined by combining the current population and the previous archive. Non-dominated sorting is then used to classify the population into a number of Pareto fronts. The first front is the best in the combined population. The archive is created by selecting fronts based on their rankings. If the number of individuals in the archive is smaller than the population size, the next front will be selected and so on. If adding a front would result in the number of individuals in the archive exceeding the initial population size, a truncation operator is applied to that front based on the crowded tournament selection by which the winner of two same rank solutions is the one that has the greater crowding distance.

MAs are a type of evolutionary algorithm and have been recognized as a powerful algorithmic paradigm on complex search spaces for evolutionary computing[25]. They apply a separate local search algorithm to improve the fitness of individuals by special hill-climbing. MAs are inspired by models of adaptation in natural systems that combine evolutionary adaptation of populations of individuals with individual learning with a lifetime. A *meme* is a unit of information that reproduces itself when people exchange ideas. Memes are adapted by the people who transmit them before being passed on to the next generation. MAs use evolutionary algorithms to perform exploration and use local searches to exercise exploitation. Local search in MAs is similar to simple hill-climbing but with the differences that (a) the neighbourhood of the current solutions is searched systematically instead of randomly in the space of all candidate solutions, and (b) the neighbourhood search is repeated until a locally optimal solution is found. An advantage of local search in MAs over other heuristics is that local exploitation around an individual can be performed much more effectively. Hence good solutions in a small region of the search space can be found quickly.

Tavakkoli-Moghaddam and Rahimi-Vahed[14] were the first pioneers in the application of MAs to multi-criteria sequencing problems for MMAL in a JIT production system. Three objectives weighted by their relative importance were optimized simultaneously. MA with a hybrid local search was proposed to generate suitable sequences. The ability of MA to find promising results was reported especially in large-sized problems. Although multi-objective functions were considered in their paper, they were combined to a single objective by the weighted sum approach. This approach is highly criticized in practice since even an experienced decision maker will still be uncertain about how much weight should be given to each objective. In addition, the relative performances of individual local searches were not examined. To answer these questions, this paper employs a Pareto-based approach rather than the weighted sum objective and also investigates the contribution of several local searches applied in several stages of different MAs.

MULTI-OBJECTIVE MIXED-MODEL SEQUENCING PROBLEM

The MMAL in this study is configured as a straight line using a conveyor system to transport parts at a constant speed. Materials to produce similar products are introduced onto the conveyor at a fixed cycle. The line is properly partitioned into m workstations and their workloads are somewhat evenly distributed as a result of line balancing. Each workstation is of closed type which means that a worker is restricted to work within its boundaries and the walking time is negligible. The task times are deterministic. The concept of a minimum part set which is a vector representing a product mix[7] is used in this paper.

The two objective functions to be simultaneously optimized are the setup times and production smoothing. Sequence-dependent setups are formulated using

$$f_1(x) = S = \sum_{k=1}^{D_\mathrm{T}} s_{[k-1],[k]}, \tag{4}$$

where S is the total setup time required for a production sequence, $s_{[k-1],[k]}$ is the setup time required if the products in position $k-1$ and k of the launch order are different, and D_T is the total demand (also represents the number of positions in a sequence). A sequence-dependent setup is needed if the product in position $k-1$ is different from the one in position k. In this study, setup times are generated from a discrete uniform distribution $U[0,100]$. Since they may affect the number of Pareto solutions obtained, the coefficient of variation (c_v) is used to measure setup time variability and $c_v = \mathrm{var}(s)/\bar{s}^2 = 1/3$ for obtaining more uniform distributed solutions.

Another crucial goal for efficient sequencing of MMAL in a JIT production system is to keep a constant consumption rate of each part on the line (production smoothing)[1] which can be formulated using

$$f_2(x) = U = \sum_{k=1}^{D_\mathrm{T}} \sum_{i=1}^{n} \left(x_{i,k} - k\frac{d_i}{D_\mathrm{T}} \right)^2, \tag{5}$$

where U is the production rates variation of a production sequence, $x_{i,k}$ is the total number of units of product i produced over stage 1 to k, $k = 1, \ldots, D_\mathrm{T}$, n is the number of unique products to be produced, and d_i is the demand for product i. U gives the matching between the actual product intermixing and demand for a production sequence as a metric to quantify this goal[2, 26].

Feasible sequences for MMAL sequencing having n products can be computed using

$$\text{Feasible Sequences} = \frac{\left(\sum_{i=1}^{n} d_i\right)!}{\prod_{i=1}^{n} (d_i!)}. \qquad (6)$$

Mansouri[13] mentioned that obtaining sequences with acceptable levels of both $f_1(x)$ and $f_2(x)$, minimum setup times, and minimum usage rate variation is NP-hard. Hence finding optimal solutions within a reasonable amount of time becomes impractical for large-sized problems.

PROPOSED MEMETIC ALGORITHM

Initialization

Initial population generation is the first step in the proposed MA (Fig. 2). A set of N chromosomes is generated randomly as an initial set of populations. The chromosome comprises a sequence of genes (products) produced in one cycle of the minimum part set (MPS) demand. The position of a gene in a sequence of the chromosome represents the product in the sequence and demonstrates model launching to MMAL. For example, if the MPS is $d = (d_A, d_B, d_C) = (2, 3, 1)$, where d_X is the number of products of type X to be produced in one production cycle, a feasible chromosome can be represented as ABABCB. Such a representation is good at preserving sequence information.

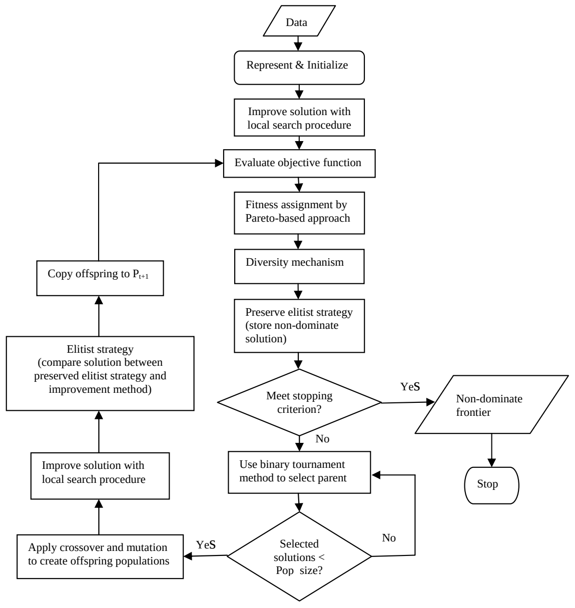
Data
Represent & Initialize
Improve solution with local search procedure
Evaluate objective function
Fitness assignment by Pareto-based approach
Diversity mechanism
Preserve elitist strategy (store non-dominate solution)
Meet stopping criterion?
YeS
Non-dominate frontier
Stop
No
Use binary tournament method to select parent
Selected solutions < Pop size?
No
YeS
Apply crossover and mutation to create offspring populations
Improve solution with local search procedure
Elitist strategy (compare solution between preserved elitist strategy and improvement method)
Copy offspring to P_{t+1}

Fig. 2 Flowchart of the proposed memetic algorithm.

Local search heuristics

Once a set of new chromosomes is created in the initial population, their performances may be improved by applying an appropriate local search. To design efficient MAs for continuous optimization, four questions related to local search executions need to be considered: (1) On which solutions should local search be applied? (2) How often should local search be applied? (3) How long should local search be run? (4) Where within the steps in the algorithm of the MA should the local search be taken?

The first question is related to the parameter P_{LS}. The solutions for applying a local search to can be chosen from all solutions, all solutions with probability P_{LS}, only first rank solutions, or first rank solutions with probability P_{LS}[27]. Applying the local search to all obtained solutions is time consuming. On the other hand, it would be too restrictive to consider only first rank solutions. Hence, we choose all solutions with probability P_{LS}. For example, $P_{\text{LS}} = 0.8$ means 80% of all solutions will be subjected to local search. In addition, if the solutions on which the local search is applied are randomly selected, the improved quality of the new solutions may not be guaranteed. Hence, to select an appropriate solution to apply the local search, binary tournament selection is used. The second question is related to the interval of generations of the MA to which the local search is applied. In this paper, the local search is applied to every generation of the MA. The third question is related to the parameter k which specifies the maximum number of examined neighbours of the current solution. The execution of local search is terminated when no better solution is found among k neighbours. If an improvement is found while generating neighbours, two improvement strategies can be used. For the first improvement strategy, the current solution is replaced immediately with the first neighbour that is better than the current solution. Alternatively, the best improvement strategy has to wait until all neighbours are generated and the current solution is replaced with the best improved solution[28]. It is clear that such early termination of the first improvement strategy can help decrease the computation time of a local search. The final question is related to the steps in the algorithm of the MA where the local search is applied which could be after the initial solution, after the new solution, after crossover, or after mutation. The number of steps to apply local search has a direct effect on the quality of solution and computation time. Hence, if computation time needs to be saved, local search should be taken at some specific steps in the algorithm of MA rather than at all steps. In this study, we perform the local search after obtaining the initial solution and after mutation since pilot experiments indicated that these two points were enough to find significantly improved solutions, pull the solutions out of the local optimum, and reduce computational time.

Here several local searches (detailed below) are evaluated to demonstrate their performance in terms of improving the efficiency of such high performance meta-heuristics as SPEA 2 and NSGA II. The local searches we use are modified from Kumar and Singh[29] to solve travelling salesman problems by repeatedly exchanging edges of the tour until no improvement is attained.

Pairwise Interchange (PI): Select two arbitrary products located at positions i and j, $i \neq j$, and interchange them to generate a neighbouring solution. All possible swaps of pairs of products in a given solution are feasible. E.g.:

Parent	ABCABCCABA
Neighbour	ABAABCCCBA

Adjacent Pairwise Interchange (API): It is a subset of PI where two products located at adjacent positions i and $i+1$ ($1 \leqslant i \leqslant n-1$) are interchanged to generate a neighbouring solution. E.g.:

Parent	AB | CA | BCCABA
Neighbour	AB | AC | BCCABA

Insertion Procedure (IP): Remove a product from one position i and then insert it back to any position j where $i \neq j$ of a given sequence. E.g.:

Parent	AB | C | ABCCABA
Neighbour	ABABCCA | C | BA

2-opt: A neighbouring solution is obtained by randomly selecting a part of the chromosome and then reversing the order of all products located in this part. E.g.:

Parent	AB | CABCCA | BA
Neighbour	AB | ACCBAC | BA

3-opt: A neighbouring solution is obtained by randomly selecting two adjacent parts of the chromosome and then reversing the orders of all products located in each part. E.g.:

Parent	AB | CAB | CCA | BA
Neighbour	AB | BAC | ACC | BA

Or-opt: It considers a smaller percentage of exchange that would be considered by a regular 3-opt by selecting a set of three adjacent products, reversing their order, and moving this reversed set to the end of the chromosome. E.g.:

Parent	AB | CAB | CCABA
Neighbour	AB | CCABA | BAC

Double-bridge (DB): It cuts the chromosome into 4 segments by deleting four random genes and then reinserts them in a different order to create a new chromosome. E.g.:

Parent	A | BC | ABC | CA | BA
Neighbour	A | CA | ABC | BC | BA

We let LS_1*LS_2 denote the local search combination applied after initial solution and after mutation, respectively. For example, IP*PI means to generate an offspring, the parent has to be modified first after initial solution by the IP operator and later after mutation by the PI operator.

Criteria to accept a move

If any of the following 4 criteria are met[30], then a neighbour solution S' is accepted and hence replaces the current solution S. (1) If the setup time of S' is less than that of S i.e., $\text{accept}(S,S') = f_1(S') - f_1(S) < 0$. (2) If the production rates variation of S' is less than that of S i.e., $\text{accept}(S,S') = f_2(S') - f_2(S) < 0$. (3) If S' dominates S $\text{accept}(S,S') = ((f_1(S') - f_1(S) \leqslant 0)(f_2(S') - f_2(S) < 0))$ or $((f_1(S') - f_1(S) < 0)$ and $(f_2(S') - f_2(S) \leqslant 0))$. (4) If both S and S' are non-dominated solutions and $\text{accept}(S,S') = w_1 \cdot (f_1(S') - f_1(S)) + (1 - w_1) \cdot (f_2(S') - f_2(S)) \leqslant 0$. The weight w_1 that determines the descent direction must be computed. The aim is to improve the front, with emphasis on the two extreme solutions, while preserving the spacing between solutions. Therefore, w_1 is defined for the current solution S by

$$w_1 = \frac{\dfrac{f_1(S) - f_1^{\min}}{f_1^{\max} - f_1^{\min}}}{\dfrac{f_1(S) - f_1^{\min}}{f_1^{\max} - f_1^{\min}} + \dfrac{f_2(S) - f_2^{\min}}{f_2^{\max} - f_2^{\min}}}, \qquad (7)$$

where $f_i^{\min}$ and $f_i^{\max}$ denote the minimum and maximum values of ith objective function in the current population, respectively.

Selection

Binary tournament selection is used to obtain suitable parents to further perform local search and genetic operators. For each chromosome, the rank on the Pareto-front is computed (SPEA2 and MSPEA 2 use strength of dominators, NSGA II and MNSGA II use non-dominated sorting). Two candidate chromosomes are chosen at random and the best individual of that set with lower rank (higher fitness) is selected as a parent. If two individuals have the same rank, the tournament prefers the most isolated one, using a density mechanism (SPEA2 and MSPEA 2 use k-nearest neighbour, NSGA II and MNSGA II use crowding distance). An important issue needs mentioning on the selection scheme that combines rank identification and density mechanism of MOEAs. The rank identification mainly emphasizes an improving solution quality (exploitation) which has no control over diversity, whereas the density mechanism focuses on the diversity of solutions (exploration). These two mechanisms are combined to round off the weakness of each other.

Crossover

The order crossover[31] was adopted as the crossover operator for all algorithms in this research. This scheme tends to maintain the relative order of the jobs. To illustrate how it works, consider the following parents.

Parent 1: ABE | CAAD | EDBCB
Parent 2: CCD | EEBA | DBBAA

A fragment of the chromosome is randomly selected as a portion of the sequences that will remain intact and become part of the offspring in the crossover process (i.e., CAAD and EEBA for parents 1 and 2, respectively). A new parent is then created by moving all the characters appearing after the end of the fragment of the original parent to the beginning of the sequence as shown below.

Parent 1′: EDBCB | ABE | CAAD
Parent 2′: DBBAA | CCD | EEBA

From these new sequences, the characters in Parent 2′ that match the characters of the fragments of the other original parent (Offspring 1: CAAD) are removed.

Offspring 1: XXX | CAAD | XXXXX
Parent 2′: XBBXX | XCD | EEBA

This mechanism applies to Offspring 2 and Parent 1′ as well. The sequence of the remaining characters in Parent 2′ is BBCDEEBA. This sequence is used to fill in the blank positions of the Offspring 1 to obtain the Offspring 1. Using this technique, the following offsprings are generated.

Offspring 1: BBC | CAAD | DEEBA
Offspring 2: DCB | EEBA | BCAAD

Mutation

Mutation is performed irregularly so that a solution could have an occasional trait that is absent from its parent. It plays an important role in preserving the diversity of the population. Mutation for this study is the swapping of two unique elements randomly selected in the sequence of interest. E.g.:

Before: CBAABDEABECD
After: CBDABDEABECA

Diversity mechanism

The diversity mechanism is exercised when many individuals of the current generation do not dominate each other and only some of them have to be selected. It calculates density information of each individual. The one with lower density has a higher chance to be selected since less non-dominated solutions are clustering around (higher diversity). The density estimation technique for SPEA 2 and MSPEA 2 is the kth nearest neighbour method[22], whereas NSGA II and MNSGA II use the crowding distance method[23].

Elitism strategy

Elitism is the mechanism of constantly updating and keeping the best solutions found so far. We use a straightforward implementation of elitism in all algorithms. An archive with a fixed number of elitists is established. The non-dominated individuals generated by the main population are considered as a set of tentative elitists. These individuals are added to the original archive. The non-dominated solutions residing in the archive are updated and the dominated ones are discarded. At each generation, a certain number of elitists are also copied into the main population to perform the local search (for MSPEA 2 and MNSGA II) and genetic operations. Therefore, by this two-way communication method, the elitist's archive is updated generation by generation and the valuable schemas of an elitist can be inherited by their offspring. For this reason, the elitism scheme has the potential to help the entire population converge into a near-optimal Pareto front.

Performance measurement

We use three metrics to assess the achievement of two goals of a multi-objective optimization as recommended by Kumar and Singh[29]: (1) convergence to the Pareto-optimal set, and (2) maintenance of diversity in the solutions of Pareto-optimal set. The convergence of the obtained Pareto-optimal solution towards a true Pareto-set (A^*) is the difference between the obtained solution set and the true-Pareto set. Mathematically, it is defined as

$$\mathrm{convergence}(A) = \frac{\sum_{i=1}^{\|A^*\|} t_i}{\|A^*\|} \tag{8}$$

where where $\|A^*\|$ is the number of elements in set A, and t_i is the Euclidean distance between the ith non-dominated solution in the true-Pareto front (y) and the obtained solutions (x) and is given by

$$t_i = \min_{j=1}^{\|A^*\|} \sqrt{\sum_{k=1}^{2} \left(\frac{f_k(x) - f_k(y)}{f_k^{\max} - f_k^{\min}} \right)^2}, \tag{9}$$

where $f_k^{\max}$ and $f_k^{\min}$ are maximum and minimum values of kth objective functions in the true-Pareto set respectively. For this metric, a lower value indicates superiority of the solution set. When all solutions converge to the Pareto-optimal front, this metric is zero indicating that the obtained solution set has all solutions in the true Pareto set. The approximated true Pareto-optimal front is the result of combining all non-dominated solutions obtained from of all four algorithms (SPEA 2, NSGA II, MSPEA 2, and MNSGA II) and recalculating a combined front.

The second measure is a spread metric. This measure computes the distribution of the obtained Pareto-solution by calculating a relative distance between consecutive solutions using

$$\text{spread}(A) = \frac{r_f + r_l + \sum_{i=1}^{\|A\|-1} |r_i - \bar{r}|}{r_f + r_l + (\|A\| - 1)\bar{r}} \tag{10}$$

$$r_i = \sqrt{\sum_{k=1}^{2} \left(\frac{f_k(x_i) - f_k(x_{i+1})}{f_k^{\max} - f_k^{\min}} \right)^2}, \tag{11}$$

where r_f and r_l are the Euclidean distances between the extreme solutions and boundary solutions of the obtained Pareto-optimal, $\|A\|$ is the number of elements in the obtained-Pareto solutions, r_i is the Euclidean distance between consecutive solutions in the obtained-Pareto solutions for $i = 1, 2, \ldots, \|A\| - 1$, and $\bar{r}$ is the average of r_i. The value of this measure is zero for a uniform distribution, but it can be more than 1 when a bad distribution is found.

The third measure is the ratio of non-dominated solutions which indicates the coverage of one set over another. Let A_j be a solution set ($j = 1, 2, \ldots, J$). For comparing each solution set ($A = A_1 \cup A_2 \ldots \cup A_J$), the ratio of non-dominated measure of the solution set A_J to the J solution sets is the ratio of solutions in A_J that are not dominated by any other solution in A, which is defined as:

$$R_{\text{NDS}}(A_j) = \frac{\|A_j - \{x \in A_j | \exists\, y \in A : y \prec x\}\|}{\|A_j\|}, \tag{12}$$

where $y \prec x$ means the obtained solution x is dominated by the true-Pareto solution y. The higher ratio indicates superiority of one solution set over another.

Experimental Design

In this paper, the relative performance of four MOEA, i.e., SPEA 2, NSGA II, MSPEA 2, and MNSGA II, are evaluated. Different local searches are added into original algorithms of SPEA 2 and NSGA II. To find an appropriate local search for our proposed MA, the same genetic parameters are used in all four algorithms including population size, elitism probability, crossover probability, inversion probability, and mutation probability which were set at 200, 1.0, 1.0, 0.5, and 0.005, respectively. For MSPEA 2 and MNSGA II, the archive size was set at 200. In the proposed MA, the local search probability (P_{LS}) is set to 0.8 and the number of neighbours to be examined (k) is 4. These settings were obtained from the results of pilot experiments. To evaluate the performance of SPEA 2, NSGA II, MSPEA 2, and MNSGA II, three measures were employed including convergence, spread, and ratio of non-dominated solutions. Each algorithm was applied to each tested problem 10 times with different initial populations. The problem sets tested in this paper were adopted from McMullen[10] as shown in Table 1. Problem sets 1–3 and 4–5 are considered as small-sized and large-sized problems, respectively.

EXPERIMENTAL RESULTS

Table 2 summarizes the performance of local searches that were applied to strengthen the original MOEAs (SPEA 2 and NSGA II). The element in the table is the local search combination that shows high or poor performances among those considered. ANOVA was conducted to test significant difference at 0.05 significant level among different local search combinations applied to MNSGA II and MSPEA 2. Those showing outstanding performances appear in Table 2. For example, under the problem set 2.2 of MNSGA II, the combination of local searches PI*IP outperforms the others. Its detailed performances in terms of convergence, spread, and ratio of non-dominated solution metrics can be seen in Table 3.

For small-sized problems (problem sets 1–3), the local searches PI, IP, and API often appear as high performance local searches. They show better performance in all aspects than the others both when applying after obtaining initial solution and after mutation. Double-bridge always shows poor convergence to true-Pareto set especially when it is applied after mutation. Interaction between local searches applied after initial solution and after mutation is found since the best combination of local searches is changed from problem to problem, e.g., PI*IP and API*IP for problem sets 2.2 and 2.3 respectively. However, it is

Table 1 Problem sets for MMAL sequencing.

	Problem set 1: 5 Product types	
	MPS	Solutions
1.1	(5,3,2,1,1)	332 640
1.2	(4,3,2,2,1)	831 600
1.3	(3,3,2,2,2)	1 663 200
	Problem set 2: 5 Product types	
	MPS	Solutions
2.1	(7,3,2,2,1)	10 810 800
2.2	(5,3,3,3,1)	50 450 400
2.3	(3,3,3,3,3)	168 168 000
	Problem set 3: 5 Product types	
	MPS	Solutions
3.1	(8,7,2,2,1)	2.993×10^{9}
3.2	(5,5,5,3,2)	1.173×10^{11}
3.3	(4,4,4,4,4)	3.055×10^{11}
	Problem set 4: 10 Product types	
	MPS	Solutions
4.1	(7,5,1,1,1,1,1,1,1,1)	4.023×10^{7}
4.2	(5,5,3,1,1,1,1,1,1,1)	2.816×10^{13}
4.3	(2,2,2,2,2,2,2,2,2,2)	2.376×10^{15}
	Problem set 5: 15 Product types	
	MPS	Solutions
5.1	(20,20,20,15,15,1,1,1,1,1,1,1,1,1,1)	3.790×10^{78}
5.2	(15,15,15,10,10,10,10,5,4,1,1,1,1,1,1)	8.357×10^{91}
5.3	(7,7,7,7,7,7,7,7,7,7,6,6,6,6,6)	6.334×10^{103}

Table 2 Performance comparison of local searches used in MNSGA II and MSPEA 2 for the various problem sets (PS).

PS	MNSGA II		MSPEA 2	
	HPLS	PPLS	HPLS	PPLS
1.1	API*2-Opt, 2-Opt*IP, Or-Opt*API	PI*Or-Opt	DB*3-Opt	2-Opt*DB
1.2	DB*IP	DB*DB	IP*IP	2-Opt*Or-Opt
1.3	DB*IP	Or-Opt*DB	Or-Opt*IP	3-Opt*DB
2.1	API*2-Opt	DB*Or-Opt	Or-Opt*API	Or-Opt*DB
2.2	PI*IP	DB*DB	3-Opt*IP	Or-Opt*DB
2.3	PI*API	2-Opt*DB	API *2-Opt	PI*DB
3.1	API*IP	DB*DB	API*IP	PI*DB
3.2	PI*IP	3-Opt*DB	API*3-Opt	API*DB
3.3	API*IP	DB*Or-Opt	2-Opt*IP	3-Opt*DB
4.1	2-Opt*API	3-Opt*DB	2-Opt*DB	Or-Opt*DB
4.2	PI*IP	API*Or-Opt	PI*IP	3-Opt*DB
4.3	Or-Opt*IP	3-Opt*DB	IP*IP	3-Opt*DB
5.1	PI*PI	2-Opt *3-Opt	PI*PI	Or-Opt* DB
5.2	PI*IP	2-Opt*2-Opt	Or-Opt*API	IP*3-Opt
5.3	PI*IP	DB*Or-Opt	2-Opt*3-Opt	2-Opt*Or-Opt

HPLS = high performance local search
PPLS = poor performance local search

Table 3 Performance comparison between MAs and original MOEAs.

PS	MNSGA II	MSPEA 2	NSGA II	SPEA 2
1.1	0.0007* (0)**	0.0017 (1)	0.0357 (0)	0.0241 (13)
	0.0177 (2)	0.0200 (5)	0.0322 (18)	0.0222 (11)
	95.65	100.00	5.56	73.68
1.2	0.0062 (1)	0.0140 (1)	0.0429 (0)	0.0244 (0)
	0.0202 (2)	0.0269 (3)	0.0232 (15)	0.0200 (5)
	90.91	84.21	0.00	24.00
1.3	0.0034 (0)	0.0071 (0)	0.0567 (1)	0.0169 (0)
	0.0257 (1)	0.0263 (2)	0.0175 (3)	0.0179 (2)
	82.61	85.00	0.00	21.43
2.1	0.0183 (0)	0.0255 (4)	0.0418 (0)	0.0315 (1)
	0.0229 (6)	0.0237 (8)	0.0287 (6)	0.0266 (18)
	66.67	45.83	0.00	7.69
2.2	0.0097 (2)	0.0175 (1)	0.0429 (0)	0.0244 (0)
	0.0199 (3)	0.0265 (6)	0.0232 (15)	0.0200 (5)
	60.00	50.00	0.00	3.45
2.3	0.0181 (0)	0.0200 (0)	0.0997 (1)	0.0336 (0)
	0.0256 (4)	0.0405 (26)	0.0343 (9)	0.0222 (6)
	57.14	53.85	0.00	33.33
3.1	0.0159 (0)	0.0233 (1)	0.0628 (0)	0.0350 (2)
	0.0200 (2)	0.0222 (5)	0.0303 (30)	0.0246 (4)
	25.00	9.09	0.00	3.85
3.2	0.0225 (0)	0.0288 (0)	0.1090 (0)	0.0441 (1)
	0.0170 (24)	0.0212 (13)	0.0352 (22)	0.0253 (7)
	5.13	12.50	0.00	8.33
3.3	0.0220 (1)	0.0301 (2)	0.2114 (7)	0.0468 (0)
	0.0253 (7)	0.0304 (17)	0.0497 (40)	0.0291 (9)
	47.06	5.26	0.00	5.26
4.1	0.0505 (4)	0.0912 (18)	0.1696 (1)	0.0738 (7)
	0.0365 (26)	0.0292 (15)	0.0293 (7)	0.0268 (14)
	25.00	0.00	0.00	0.00
4.2	0.0299 (0)	0.0470 (0)	0.2068 (6)	0.0701 (0)
	0.0254 (13)	0.0377 (27)	0.0323 (7)	0.0329 (13)
	20.83	0.00	0.00	0.00
4.3	0.1158 (4)	0.1933 (11)	0.4680 (22)	0.1823 (5)
	0.0481 (42)	0.0507 (19)	0.0828 (68)	0.0360 (9)
	20.00	0.00	0.00	0.00
5.1	0.0584 (0)	0.1051 (1)	0.3817 (31)	0.0845 (1)
	0.0261 (6)	0.0251 (15)	0.0269 (7)	0.0257 (6)
	41.67	0.00	0.00	0.00
5.2	0.0299 (1)	0.1473 (4)	0.4059 (31)	0.0835 (0)
	0.0214 (9)	0.0327 (8)	0.0295 (8)	0.0230 (5)
	47.50	0.00	0.00	0.00
5.3	0.0915 (1)	0.1025 (1)	0.5113 (31)	0.2043 (2)
	0.0292 (10)	0.0341 (42)	0.0312 (7)	0.0306 (34)
	47.50	0.00	0.00	0.00

For each problem set, first row = convergence, second row = spread, third row = ratio of solution (%)
* = mean; ** = variance $\times 10^{-4}$

noticeable that IP often shows high performance when applied after mutation.

For large-sized problems (problem sets 4–5), PI and IP still show outstanding performances, whereas the local search that performs poorly is still the double-bridge especially when it is applied after mutation. Often, IP still shows better performance when it is applied after mutation. Interaction between local searches applied after initial solution and after mutation is still noticeable. Notice that PI*IP appears very often as the best contestant.

Table 3 illustrates the relative performances of MNSGA II, MSPEA 2, NSGA II, and SPEA 2 as gauged by the convergence, spread, and ratio of non-dominated solution metrics. A lower value of the convergence metric is better, whereas higher values of spread and ratio of non-dominated solution metrics are better. It is obvious that SPEA2 outperforms NSGA II for convergence and ratio of non-dominated solution metrics. For spread metric, the value obtained from SPEA 2 and NSGA II is not substantially different. This means the non-dominated fronts of SPEA 2 and NSGA II are more or less evenly distributed. Similarly, the spread metrics of MSPEA 2 and MNSGA II are quite similar and are not much different from those of the non-MA algorithms. This means the performance of diversity mechanisms of MA algorithms and non-MA algorithms are not significantly different.

When the appropriate local search is applied to MAs, their performances are improved significantly. For example, under the problem set 2.3, the ratio of non-dominated solutions increases from 0.00 (no non-dominated solution found) for NSGA II to 57.14% for MNSGA II. As a result, it is obvious that MAs (MSPEA 2 and MNSGA II) with appropriate local searches can help improve the performances of such original MOEAs as SPEA 2 and NSGA II.

With small-sized problems (problem sets 1–3), the performance differences between MNSGA II and MSPEA 2 are not substantial. However, when the problem sizes are large, MNSGA II outperforms MSPEA 2 for convergence and ratio of non-dominated solution metrics. Such results contradict those of the original MOEAs (SPEA 2 outperforms NSGA II). As a result, it is clear that the appropriate local search can enhance the performances of NSGA II more than SPEA 2. In addition, the viable MA is MNSGA II.

CONCLUSION

In this paper, an attempt was made to study the performance of MAs to search for Pareto-optimal solutions of MMAL sequencing problems in a JIT environment where the objectives of dependent setup times and production rate variation are considered simultaneously. Since the objectives may conflict with each other, a sequence that can optimize both objectives at the same time may not exist. Furthermore, this type of problem is NP-hard, and so obtaining desirable solutions through MOEAs is a practical option. The basic concept of our MAs is to combine original MOEAs with appropriate local searches. Several local searches are evaluated as to whether they can work well with original MOEAs (MSPEA 2 and MNSGA II). The first improvement is the strategy that we employ to reduce the computation time of the local search operation. Under this strategy, the neighbours of the current solution are accepted right after finding a better solution and terminated when no better solution is found among k neighbours randomly generated from the current solution. Moreover, local search is not applied to all the selected solutions but with probability P_{LS} for decreasing the number of solutions to apply local search. We found that MNSGA II and MSPEA 2 outperform the original MOEAs. MNSGA II shows better performances than MSPEA 2, whereas SPEA 2 outperforms NSGA II. Local searches PI and IP perform better than the others. Comparing the performances of some appropriate local search, in practice, decision makers should give high priority to the one that can give significant improvement on performances and consumes less computation time. Fortunately, in this study, both PI and IP local searches perform well with a short CPU time. However, if the situation is not the same as in this study, decision makers have to trade off between time (cost) and quality of solutions which depend on which goal is much more important at that time.

REFERENCES

1. Mondon Y (1983) *Toyota Production System*, Institute of Industrial Engineering Press, Atlanta.
2. Miltenburge J (1989) Level schedules for mixed-model assembly lines in just-in-time production systems. *Manag Sci* **35**, 192–207.
3. Miltenburg J, Steiner G, Yeomans S (1990) A dynamic programming algorithm for scheduling mixed-model just-in-time production systems. *Math Comput Model* **13**, 57–66.
4. Kubiak W, Sethi SP (1991) A note on level schedules for mixed-model assembly lines in just-in-time production systems. *Manag Sci* **37**, 241–59.
5. Korkmazel T, Meral S (2001) Bi-criteria sequencing methods for the mixed-model assembly line in just-in-time production systems. *Eur J Oper Res* **131**, 188–207.
6. Bard JF, Shtub A, Joshi SB (1994) Sequencing mixed-model assembly lines to level parts usage and minimize the length. *Int J Prod Res* **32**, 2431–54.
7. Hyun CJ, Kim Y, Kim YK (1998) A genetic algorithm for multiple objective sequencing problems in mixed model assembly lines. *Comput Oper Res* **21**, 675–90.
8. McMullen PR (1998) JIT sequencing for mixed model assembly lines with setups in Tabu search. *Prod Plann Contr* **9**, 504–10.
9. McMullen PR, Frazier GV (2000) A simulated annealing approach to mixed-model sequencing with multiple objectives on a JIT line. *IIE Trans* **32**, 679–86.
10. McMullen PR (2001) An efficient frontier approach to addressing JIT sequencing problems with setups via search heuristics. *Comput Ind Eng* **41**, 335–53.
11. McMullen PR (2001) A Kohonen self-organizing map approach to addressing a multiple objective, mixed-

model JIT sequencing problem. *Int J Prod Econ* **72**, 59–71.

12. McMullen PR (2001) An ant colony optimization approach to addressing a JIT sequencing problem with multiple objectives. *Artif Intell Eng* **15**, 309–17.
13. Mansouri SA (2005) A multi-objective genetic algorithm for mixed-model sequencing on JIT assembly lines. *Eur J Oper Res* **3**, 696–716.
14. Tavakkoli-Moghaddam R, Rahimi-Vahed AR (2006) Multi-criteria sequencing problem for a mixed-model assembly line in a JIT production system. *Appl Math Comput* **181**, 1471–81.
15. 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 Informat* **21**, 85–9.
16. Moscato P (1999) Memetic algorithms: A short introduction. In: Corne D, Dorigo M, Glover F (eds) *New Ideas in Optimization*, vol 14, McGraw-Hill, London, pp 219–34.
17. Konak A, Coit DW, Smith AE (2006) Multi-objective optimization using genetic algorithms: A tutorial. *Reliab Eng Syst Saf* **91**, 992–1007.
18. Coello CAC, Veldhuizen DA, Lamont GB (2002) *Evolutionary Algorithms for Solving Multi-Objective Problems*, Kluwer, Dordrecht.
19. Zitzler E, Thiele L (1999) Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach. *IEEE Trans Evol Comput* **3**, 257–71.
20. Fonseca CM, Fleming PJ (1993) Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization. In: Proceedings of 5th International Conference on Genetic Algorithms, vol 17-23, pp 416–23.
21. Fonseca CM, Fleming PJ (1995) An overview of evolutionary algorithms in multiobjective optimization. *Evol Comput* **3**, 1–16.
22. Zitzler E, Laumanns M, Thiele L (2001) SPEA 2: Improving the strength Pareto evolutionary algorithm. TIK Report 103, Computer Engineering and Networks Laboratory, Swiss Federal Institute of Technology.
23. Deb K, Pratap A, Agarwal S, Meyarivan T (2002) A fast and elitist multiobjective genetic algorithm: NSGA II. *IEEE Trans Evol Comput* **6**, 182–97.
24. Zitzler E, Thiele L, Deb K (2000) Comparison of multiobjective evolutionary algorithms: Empirical results. *Evol Comput* **8**, 173–95.
25. Corne D, Dorigo M, Glover F (1999) *New ideas in optimization*. McGraw-Hill, London.
26. Miltenburg J, Sinnamon GT (1989) Scheduling mixed-model multi-level just-in-time production systems. *Int J Prod Res* **27**, 1487–509.
27. Lacomme P, Prins C, Sevaux M (2004) A genetic algorithm for a bi-objective capacitated arc routing problem. *Comput Oper Res* **33**, 3473–93.
28. Ishibuchi H, Yoshida T, Murata T (2003) Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling. *IEEE Trans Evol Comput* **7**, 204–23.
29. Kumar R, Singh PK (2007) Pareto evolutionary algorithm hybridized with local search for biobjective TSP. In: Grosan C, Abraham A, Ishibuchi H (eds) *Hybrid Evolutionary Algorithms*, Springer, Berlin, pp 361–98.
30. Lacomme P, Prins C, Sevaux M (2004) A genetic algorithm for a bi-objective capacitated arc routing problem. *Comput Oper Res* **33**, 3473–93.
31. Michalewicz Z (1996) *Genetic Algorithms + Data Structures = Evolution Programs*, Springer, Berlin.