Available online at www.sciencedirect.com

SCIENCE DIRECT®

European Journal of Operational Research 167 (2005) 696–716

EUROPEAN JOURNAL OF OPERATIONAL RESEARCH

www.elsevier.com/locate/dsw

A Multi-Objective Genetic Algorithm for mixed-model sequencing on JIT assembly lines

S. Afshin Mansouri [*]

Industrial Engineering Department, Amirkabir University of Technology, P.O. Box 15875-4413, Tehran, Hafez Ave., Iran

Available online 9 September 2004

Abstract

This paper presents a Multi-Objective Genetic Algorithm (MOGA) approach to a Just-In-Time (JIT) sequencing problem where variation of production rates and number of setups are to be optimized simultaneously. These two objectives are typically inversely correlated with each other, and therefore, simultaneously optimization of both is challenging. Moreover, this type of problem is NP-hard, hence attainment of IP/LP solutions, or solutions via Total Enumeration (TE) is computationally prohibitive. The MOGA approach searches for locally Pareto-optimal or locally non-dominated frontier where simultaneous minimization of the production rates variation and the number of setups is desired. Performance of the proposed MOGA was compared against a TE scheme in small problems and also against three other search heuristics in small, medium and large problems. Experimental results show that the MOGA performs very well when compared against TE in a considerably shorter time. It also outperforms the comparator algorithms in terms of quality of solutions at the same level of diversity in reasonable amount of CPU time.
© 2004 Elsevier B.V. All rights reserved.

Keywords: Mixed-model sequencing; Just-In-Time; Pareto-optimal frontier; Multi-Objective Genetic Algorithms

1. Introduction

Mixed-model assembly lines are a type of production line where a variety of product models similar in product characteristics are assembled. Such an assembly line is increasingly accepted in industry to cope with the trend of diversification of customer demand. Determining the sequence of introducing models to the mixed-model assembly lines in a JIT production system is of particular importance considering the goals crucial for efficient implementation of JIT. These goals are basically: (1) leveling the load on each process within the line, and (2) keeping a constant speed in consuming each part on the line (Monden,

[*] Fax: +98 21 641 3025.
E-mail address: samansouri@hotmail.com

0377-2217/$ - see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.ejor.2004.07.016

1993). Toyota Corporation developed the Goal Chasing I and II (GC-I and GC-II) methods to handle these problems (Monden, 1993). GC-I selects a model that minimizes the one-stage variation at each stage, and GC-II simplifies GC-I under special assumptions regarding product structures. Here a "stage" represents a position in the order of a sequence.

Miltenburg (1989) formulated the mixed-model sequencing problem as a nonlinear integer programming problem with the objective to minimize the total deviation of actual production rates from the desired production rates. Under the assumption that all models require the same number and mix of parts, Miltenburg pointed out that minimizing the variation in production rates of the finished products (the product-level problem) achieves minimizing the variation in part usage rates (the part-level problem). He presents an exact algorithm to solve the program that obtains a non-integer solution and uses enumeration of partial sequences to turn it into a feasible integer one. In consequence the algorithm has exponential, in the number of products, worst case complexity. Therefore, two heuristics are proposed for the problem and limited computational experiments to compare their relative performance are carried out. Miltenburg et al. (1990) proposed a dynamic programming algorithm to solve the problem. Though exponential in the number of products, the algorithm enabled to test the performance of the scheduling heuristics proposed in Miltenburg (1989) for many manufacturing problems with large demands and small number of products. The algorithm proves to be too time consuming for very large problems. Miltenburg and Goldstein (1991) presented a dual-goal problem to incorporate both goals 1 and 2 in sequencing mixed-model assembly lines. A one-stage simplification for the station-time variation as well as a two-stage heuristic based on direct enumeration was presented to solve the dual-goal problem. The one-stage simplification by Miltenburg and Goldstein represents a significant improvement over GC-II because it does not require GC-II's assumptions regarding product structures. Kubiak and Sethi (1991) developed an optimization algorithm for the problem and its extensions that runs in polynomial time in the total demand for all products produced on the line over a given time horizon. They show that the objective function may actually be represented by using penalties for deviation from most even, perhaps unrealizable, distribution of production during a specified time horizon, and if these penalty functions are non-negative and convex, then the problem can be reduced to an assignment problem. Steiner and Yeomans (1993) developed a bi-partite model to minimize the maximal variation at the product level. Cheng and Ding (1996) presented the modifications of several existing product-level solution approaches to consider various weights on different models. Kubiak et al. (1997) developed a dynamic programming approach to obtain optimal solutions for the multi-level model in sequencing mixed-model assembly lines. It was shown that the multi-level problem is NP-hard in the strong sense. Zhu and Ding (2000) considered the two-stage part-level variation as a heuristic solution approach and discussed the effect of minimizing the product level variation on minimizing the part-level variation. A transformed two-stage solution approach for the part-level problem was presented and shown to be advantageous over the one-stage method in solution quality.

Sequencing mixed-model assembly lines have also been considered as a multi-objective problem over the past 10 years. Bard et al. (1994) developed a mathematical model that involved two objective functions: minimizing the overall line length and keeping a constant rate of part usage. They combined the two objectives using a weighted sum and suggested a Tabu Search (TS) algorithm. Tsai (1995) applied an exact optimization approach to a bi-criterion model consisting the dual objectives of minimizing the risk of line stop and minimizing total utility work. The author, however, considered small-sized problems involving two product models and one station. Hyun et al. (1998) considered three objectives: minimizing total utility work, keeping a constant rate of part usage and minimizing total setup cost and developed a genetic algorithm to solve the model. McMullen (1998) considered two objectives for minimizing the parts usage rate, as defined by Miltenburg (1989), and minimizing number of setups. He employed weighting approach to tackle the two objectives simultaneously and developed a TS approach for it. The bi-objective model of McMullen (1998) was further considered using different meta-heuristic solution approaches. McMullen and Frazier (2000) developed a Simulated Annealing (SA) for the problem and compared it against the TS approach.

They conclude that the SA approach provides superior results compared to the TS approach. McMullen et al. (2000) developed a Genetic Algorithm (GA) for the same model and compared it against the TS and SA approaches. They show that performance of TS is not well in comparison to SA and GA across the attributes of make-span and usage rate. They also present that GA and SA are equally well performing, although GA approach is computationally more expensive than SA. A comparison of three search heuristics, i.e. GA, SA and TS in finding locally efficient frontier of 45 problem instances is reported by McMullen (2001a). Application of a Kohonen Self-Organizing Map (SOM) approach was also reported in McMullen (2001b). The author concludes that the SOM approach provides near-optimal solutions in terms of the two objectives. He further adds that the SOM performs poorly with regard to CPU time. McMullen (2001c) developed an Ant Colony Optimization (ACO) approach and compared it against the SA, TS, GA and SOM approaches using the same data sets as in McMullen (2001b). His experiments show that the ACO approach is competitive with the other search heuristics in terms of performance and CPU requirements.

In this research, a MOGA approach is proposed to the multi-objective mixed-model sequencing problem addressed first in McMullen (1998). The objectives are (1) minimizing the number of required setups, and (2) minimizing variation of production rates. A setup is required each time two consecutive items in the production sequence are different. Smoothing variation of production rates is considered as a substitute for the ultimate objective of smoothing material (parts) usage rates under the assumption, as stated by Miltenburg (1989), that products require approximately equal number and mix of parts. Zhu and Ding (2000) elaborate on the effect of minimizing the product-level variation on minimizing the part-level variation and provide conditions under which the assumption is true.

The usage rate is a measure of the company's ability to keep the schedule level, or evenly inter-mixed, by keeping the raw materials for the different products arriving at the system at as constant a rate as possible. Because JIT systems are concerned with having the right parts at the right place at the right time, sequencing must be done so that raw materials are introduced into the system at a fairly uniform rate. With both the number of setups and variation of production rates, smaller values are more desirable. Production schedules or sequences that provide a strong level of product inter-mixing clearly will require more setups.

The remaining sections of the paper are organized as follows. Section 2 gives a brief explanation of multi-objective optimization. The multi-objective mixed-model sequencing problem is formulated and discussed in Section 3. Comprehensive explanation of the proposed MOGA approach is given in Section 4 followed by discussion of computational experiments in Section 5. Finally, concluding remarks are outlined in Section 6.

2. Multi-objective optimization

A Multi-objective Optimization Problem (MOP) can be defined as determining a vector of design variables within a feasible region to minimize a vector of objective functions that usually conflict with each other. Such a problem takes the form:

$$\begin{array}{ll}\text{Minimize} & \{f_1(\mathbf{X}), f_2(\mathbf{X}), \ldots, f_m(\mathbf{X})\} \\ \text{subject to} & \mathbf{g}(\mathbf{X}) \leqslant 0,\end{array} \tag{1}$$

where $\mathbf{X}$ is vector of decision variables; $f_i(\mathbf{X})$ is the ith objective function; and $\mathbf{g}(\mathbf{X})$ is constraint vector. A decision vector $\mathbf{X}$ is said to *dominate* a decision vector $\mathbf{Y}$ (also written as $\mathbf{X} \succ \mathbf{Y}$) iff:

$$f_i(\mathbf{X}) \leqslant f_i(\mathbf{Y}) \quad \text{for all } i \in \{1, 2, \ldots, m\} \tag{2}$$

and

$$f_i(\mathbf{X}) < f_i(\mathbf{Y}) \quad \text{for at least one } i \in \{1, 2, \ldots, m\}. \tag{3}$$

All decision vectors that are not dominated by any other decision vector are called *non-dominated* or *Pareto-optimal*. These are solutions for which no objective can be improved without detracting from at least one other objective.

There are various solution approaches for solving the MOP. Among the most widely adopted techniques are: sequential optimization, ε-constraint method, weighting method, goal programming, goal attainment, distance based method and direction based method. For a comprehensive study of these approaches, readers may refer to Szidarovsky et al. (1986).

Evolutionary Algorithms (EAs) seem particularly desirable to solve multi-objective optimization problems because they deal simultaneously with a set of possible solutions (the so-called population) which allows to find an entire set of Pareto-optimal solutions in a single run of the algorithm, instead of having to perform a series of separate runs as in the case of the traditional mathematical programming techniques. Additionally, EAs are less susceptible to the shape or continuity of the Pareto-optimal frontier, whereas these two issues are a real concern for mathematical programming techniques. However, EAs usually contain several parameters that need to be tuned for each particular application, which is in many cases highly time consuming. In addition, since the EAs are stochastic optimizers, different runs tend to produce different results. Therefore, multiple runs of the same algorithm on a given problem are needed to statistically describe their performance on that problem. These are the most challenging issues with using EAs for solving MOPs. For detail discussion on application of EAs in multi-objective optimization, see Coello et al. (2002) and Deb (2001).

Scheduling with multiple objectives has been one of the most attractive applications of MOP in manufacturing as well as service industries. There are numerous cases where scheduling problems deal with more than one, often conflicting objectives. For instance: project scheduling (minimizing the project's completion time besides resource leveling), flow-shop scheduling (minimizing make-span, total tardiness and total flow time), examination time-tabling (minimizing total number of violations of each type of constraints) and JIT sequencing (minimizing total utility work, keeping a constant rate of part usage, and minimizing set-up cost). For a comprehensive survey on theory and applications of multi-criteria scheduling, readers may refer to T'kindt and Billaut (2002).

3. The multi-objective mixed-model sequencing problem

3.1. Objective functions

The following variables are defined for the multi-objective JIT sequencing of mixed-model lines:

U	production rates variation of a production sequence
S	number of setups in a production sequence
a	number of unique products to be produced
D_T	total number of units for all products or total demand—also represents number of positions in sequence
d_i	demand for product i, $i = 1, 2, \ldots, a$
s_k	1 if setup is required at stage k; 0 if not
$x_{i,k}$	total number of units of product i produced over stages 1 to k, $k = 1, 2, \ldots, D_T$

The MOP for the mixed-model JIT sequencing problem can be stated as

$$\text{Minimize } \{f_1(\mathbf{X}), f_2(\mathbf{X})\}, \tag{4}$$

where $\mathbf{X}$ is the vector of decision variables, i.e. the sequence at which multiple models are introduced to the line, and:

$$f_1(\mathbf{X}) = S = 1 + \sum_{k=2}^{D_T} s_k, \tag{5}$$

$$f_2(\mathbf{X}) = U = \sum_{k=1}^{D_T} \sum_{i=1}^{a} \left(x_{i,k} - k \times \frac{d_i}{D_T} \right)^2. \tag{6}$$

Eq. (5) calculates required setups (S) in a production sequence. If the product in position k is different from the product in position $k-1$, then a setup is required and $s_k = 1$. Otherwise, a setup is not required and $s_k = 0$. It is assumed here that an initial setup is required regardless of sequence. Eq. (6) is Miltenburg's usage rate metric (U), which returns variation of production rates in a production sequence.

3.2. Non-dominated frontier

In general, lower production rates variation could be achieved by increasing number of setups. However, it is not always desirable to obtain reduced production rates variation at the expense of large setups, which may be costly even infeasible when other production and/or market demand constraints are taken into account. Thus, a solution approach would be desirable that provides the decision-makers with Pareto-optimal or non-dominated solutions, thereby enabling them to decide on the most appropriate sequence.

As an example, consider a situation where 5 products as A, B, C, D and E are to be put into production sequence ($a = 5$), where demand for the items is as follows: $d_1 = 6$, $d_2 = 3$, $d_3 = 1$, $d_4 = 1$ and $d_5 = 1$ resulting a total demand of $D_T = (6+3+1+1+1) = 12$. Consider two possible solutions for this problem as $\mathbf{X}_1 = \text{BBBCAAAAAAED}$ and $\mathbf{X}_2 = \text{EAAAAAACBBBD}$. From Eq. (5), the number of setups for these solutions (S) is equal to 5. However, based on Eq. (6), the $\mathbf{X}_1$'s production rates variation (U) is 40.83 while the same figure for $\mathbf{X}_2$ is 44.33. Hence $\mathbf{X}_1$ is preferred to $\mathbf{X}_2$. In other words, $\mathbf{X}_1$ dominates $\mathbf{X}_2$ ($\mathbf{X}_1 \succ \mathbf{X}_2$). For examining the situation with an increased setups, consider two other solutions as $\mathbf{X}_3 = \text{ABACADEABABA}$ and $\mathbf{X}_4 = \text{AEABACABDABA}$. Both solutions require 12 setups whereas the production rates variation associated with $\mathbf{X}_3$ and $\mathbf{X}_4$ is 7.67 and 8.83, respectively. Therefore, $\mathbf{X}_3$ dominates $\mathbf{X}_4$ ($\mathbf{X}_3 \succ \mathbf{X}_4$). On the other hand, although $\mathbf{X}_3$ has a lower production rates variation than $\mathbf{X}_1$, it cannot dominate $\mathbf{X}_1$, as the later solution requires less setups.

As a result, non-dominated or Pareto-optimal frontier that provides the decision-maker an opportunity to find the sequences having minimal values of production rates variation for each associated required number of setups is to be sought. Fig. 1 illustrates the non-dominated frontier found via total enumeration for the above example problem.

3.3. Combinatorial complexity

Finding production sequences with desirable levels of both number of setups and production rates variation is NP-hard as pointed out by McMullen (2001b). Total number of sequences for a mixed-model sequencing problem having a products can be computed using the general formula to compute the number of permutations of a multi-set (Walpole and Myers, 1985) as follows:

$$\text{Total sequences} = \frac{\left(\sum_{i=1}^{a} d_i \right)!}{\prod_{i=1}^{a} (d_i!)}. \tag{7}$$

As the problem increases in size, the number of feasible solutions increases in an exponential fashion, thereby attainment of optimal solutions becomes impractical for large problems. Problems with a large number of possible solutions usually cannot be solved to optimality within a reasonable amount of time.

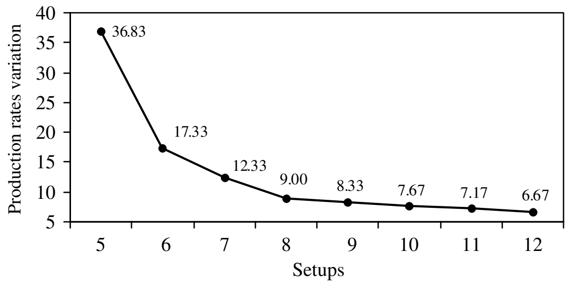

Fig. 1. Non-dominated (Pareto-optimal) frontier for example problem.

4. The proposed MOGA

As GAs work with a population of solutions, a number of locally Pareto-optimal solutions may be captured using them. Two important issues in designing MOGAs are the capabilities of exploitation and wide exploration of the search space (Hyun et al., 1998). Exploitation is the ability to find good solutions quickly in terms of Pareto optimality, and exploration is to maintain a set of diverse individuals in a population. These two are primarily controlled by the selection scheme used based on assigning appropriate fitness to the individuals in a population so that locally Pareto-optimal solutions have high chances to survive as the generations evolve. In the proposed MOGA, these have been achieved basically by using *non-dominated sorting* in conjunction with *niching*. An *elitist* selection strategy is employed to select individuals from a population to form the next population. Selected individuals are then recombined using genetic operators, i.e. *crossover*, *inversion* and *mutation*.

The general framework of the proposed MOGA is described in the following pseudocode:

```
Initialize Search Parameters
Randomly Generate Initial Solutions
i=1
Do
   Perform non-dominated Sorting
   Perform Niching
   For j=1 to Population_Size
      Calculate Dummy_Fitness_Value (chromosome_j)
   Next j
   Do
      Select chromosome to form Mating_Pool using RSSWR_UE scheme
   While (Selected_Chromosomes < Population_Size)
   Shuffle Mating_Pool
   Produce offspring using: Reproduction, Crossover, Inversion and Mutation
   i=i + 1
Until Stopping Criteria Satisfied
Combine locally non-dominated Frontier of the last Generation with Elite Set
Remove Duplicated Solutions
Report Resultant Locally non-dominated Frontier
```

Some steps of the algorithm are discussed in more details in the following sub-sections.

4.1. Non-dominated sorting

The idea behind the *non-dominated sorting* procedure is that a ranking method is used to emphasize good solutions and a *niche* method is used to maintain stable subpopulations of good solutions. Non-dominated sorting, adapted from Srinivas and Deb (1994), is carried out in a way that the population is ranked on the basis of an individual's non-domination. The locally non-dominated individuals present in the population are first identified from the current population. Then, all these individuals are assumed to constitute the first locally non-dominated frontier in the population and assigned a large *Dummy_Fitness_Value*. The same fitness value is assigned to give an equal reproductive potential to all these locally non-dominated individuals. To maintain diversity in the population, these classified individuals are then *shared* with their dummy fitness values. Sharing is achieved by performing selection operation using degraded fitness values that are obtained by dividing the original fitness value of an individual by a quantity proportional to the number of individuals exist in its niche. This causes multiple locally Pareto-optimal solutions to co-exist in the population. To calculate the niche dimensions in a given population, the concept of *niche cubicle* proposed by Hyun et al. (1998) was primarily adopted. A niche cubicle for a given individual is a rectangular region whose center is that particular individual. Hyun et al. (1998) suggest the following equation to compute size of the niche cubicle in a problem having m objectives:

$$\sigma_{lt} = \frac{Max_{lt} - Min_{lt}}{\sqrt[m]{Pop\_Size}}, \quad l = 1, 2, \ldots, m \tag{8}$$

where Max_{lt} and Min_{lt} are the maximum and the minimum of the lth objective function at generation t, and *Pop_Size* is the size of population. The niche size is calculated at every generation. Fig. 2 illustrates the construction of niche cubicles in a two-objective problem wherein two niche cubicles are shown for two arbitrarily chosen individuals $\mathbf{X}_1$ and $\mathbf{X}_2$. Since the size of every niche cubicle is the same, the solution density of a niche cubicle can be simply measured by the number of individuals included in the cubicle. A solution located in a less dense cubicle is allowed to have a higher probability to survive in the next generation. For example, the niche cubicle of $\mathbf{X}_1$ is less dense than that of $\mathbf{X}_2$, consequently $\mathbf{X}_1$ will have a higher survival probability than $\mathbf{X}_2$.

After sharing, these locally non-dominated individuals are ignored temporarily to process the rest of the population in the same way to identify individuals for the second locally non-dominated frontier. These locally non-dominated solutions are then assigned a new dummy fitness value that is kept smaller than the minimum shared dummy fitness of the previous frontier. This process is continued until the entire population is classified into several frontiers.

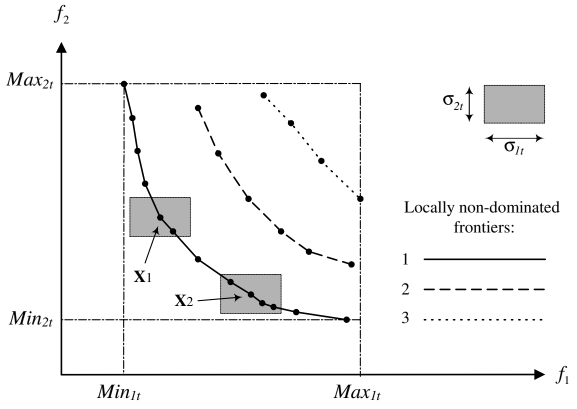

Fig. 2. Niche cubicles along locally non-dominated frontier.

4.2. Selection

A number of individuals are selected according to their fitness to form the next generation. This is carried out using a novel scheme, called *RSSWR_UE*, which is a combination of the so-called *Reminder Stochastic Sampling Without Replacement* (Goldberg, 1989) and an elitist strategy. The pseudocode for the selection scheme to select chromosomes to form the *Mating_Pool* is given in Appendix A.

4.3. Crossover

The *order crossover* (OX–Michalewicz, 1996) was selected for the MOGA. To illustrate how it works, consider the following two parent sequences:

Parent 1 : A A A A | A A B B B | B C C D D

Parent 2 : D A B A | B C B A A | B C A D A

Brackets designate the portion of the sequences that will remain intact and become part of the offspring in the crossover process. The location of the brackets is randomly determined, but the left bracket must be to the right of the first character in a sequence, and the right bracket must be to the left of the last character in the sequence. A new parent is then created by moving all the characters appearing after the right bracket (of the original parent) to the beginning of the sequence. The results of this are shown below:

Parent 1′ : **B** **C** C D | D **A** **A** A A | A A **B** B B(C D D A A A A B B)

Parent 2′ : **B** C **A** D | **A** D A **B** A | **B** C B A A(C D D A A C B A A)

From this new sequence, characters that match the characters between the brackets of the other original parent are removed (the characters to be removed are shown in bold). For example, from Parent 1′, the first two A's, the first two B's, and the first C are removed because Parent 2 had the sequence BCBAA between its brackets. The sequence between the brackets of the original parent and this shortened list (shown in parentheses above) from the other parent is then used to construct an offspring. For example, the sequence AABBB is taken from Parent 1, and the shortened list from Parent 2′ is added, starting at the right bracket and wrapping around to the beginning of the sequence. Using this technique, the following offspring result:

Offspring 1 : C B A A | A A B B B | C D D A A

Offspring 2 : A A B B | B C B A A | C D D A A

Notice how some of the characters of each offspring match that of their parents. There are other crossover techniques available for general sequencing problems, e.g. *partially-mapped crossover* (PMX; Goldberg and Lingle, 1985), *cycle crossover* (CX; Oliver et al., 1987) and *immediate successor relation crossover* (ISRX; Hyun et al., 1998).

4.4. Inversion

Inversion is an operator that generates offspring from a single parent. It first chooses two random cut points in a parent. The elements between the cut points are then reversed. An example of the inversion operator is presented below:

Before inversion : A B B | B A C C | C C

After inversion : A B B | C C A B | C C

4.5. Mutation

Mutation is sometimes performed so that a solution has an occasional trait that is unique from its parents. This is essentially done so that diversity remains in the population. Mutation for this research is the simple swapping of two unique elements in the sequence of interest. Consider the following sequence:

$$\text{Before mutation}: \text{A A A } \underline{\text{A}} \text{ A A B B B } \underline{\text{B}} \text{ C C D D}$$

The two underlined elements are randomly selected unique elements that are targeted for swapping. After swapping, or mutation, the sequence is as follows:

$$\text{After mutation}: \text{A A A } \underline{\text{B}} \text{ A A B B B } \underline{\text{A}} \text{ C C D D.}$$

5. Computational experiments

5.1. Comparator algorithms

Performance of the developed MOGA was compared against a TE scheme in small problem sets. It was also compared against three search heuristics developed for the multi-objective mixed-model sequencing problem discussed in this paper in small, medium and large problems. The comparator heuristics include: GA, SA and TS approaches addressed in McMullen (2001a).

5.2. Test problems

In order to compare the MOGA against the TE scheme as well as the three comparator heuristics, the problems used in McMullen (2001a) were selected. These include 5 problem sets, each set consist of 9 problems, as presented in Appendix B. They cover a diverse set of mixed-model sequencing problems, from the smallest problem with 11,880 solutions to the largest one with 6.334×10^{103} possible solutions.

5.3. Parameter setting

For tuning the MOGA for the mixed-model sequencing problems, extensive experiments were conducted with differing sets of parameters in a competence against the comparator algorithms. At the end, the following set was found to be effective in terms of quality as well as diversity of solutions when compared against the comparator algorithms. It should be noted that changing these parameters may result in different outcomes than those achieved in this research. The *Population_Size*, *Mutation_Rate* and *Elitism_Probability* were set to 500, 0.005 and 1.0, respectively. Regarding the elitism employed in the selection scheme (described in Appendix A), *Initial_Transfer_Probability* and *Degrading_Factor* were set, respectively to 0.1 and 0.8. Size of the elite set was set to 200. The upper bound for maximum number of generations or *Max_Gen* was set to 1000. Execution of the algorithm terminates once the number of generations reaches to this limit or alternatively, when a robust locally non-dominated frontier has been achieved. The robustness of a frontier at a given generation t is judged based on mutual comparisons of the locally non-dominated frontiers of the G consecutive generations prior to t. For the current research, G was set to 30.

For setting *Crossover_Rate* and *Inversion_Rate*, it was found that fixed rates do not result in quality as well as diverse frontiers at the same time. High rates were observed to be effective on more diversification of solutions along the frontier. However, the rate of improvement becomes constantly slower and slower until it ceases to improve amid generations. In the meantime, quality solutions could hardly be obtained using the high rates. The low rates for crossover and mutation were found to be effective on improving quality of solutions in a less-diverse frontier. To exploit advantages of both scenarios, a changing rate for the operators was considered in a way that high rates are allowed at the beginning followed by lower rates thereafter. Conducting several experiments on various schemes for changing rates for these two operators, the following formula was found to be promising to determine R_t, i.e. the rate at generation t based on an initial '*IR*' rate:

$$R_t = \begin{cases} IR, & t < (K \times Max\_Gen), \\ IR \times \dfrac{1}{t + 1 - (K \times Max\_Gen)}, & (K \times Max\_Gen) \leqslant t < Max\_Gen, \end{cases} \tag{9}$$

where K is a constant that, after some trials, was set to 0.33. Initial rates for crossover and inversion operators were set to 0.6 and 0.8, respectively. To determine the *Niche Dimensions*, Eq. (8) proposed by Hyun et al. (1998) was primarily employed. It was observed that reduced dimensions obtained by applying a multiplier $D < 1.0$ as $(D \times \sigma_{It})$ enhances quality and diversity to some extent at the same time. The multiplier D was set to 0.1, except for problem set 5, wherein a smaller multiplier of 0.002 was found to be more effective.

5.4. Hardware and software specifications

The MOGA algorithm and the TE scheme were coded in C++ and executed on a Pentium IV processor at 2.0 GHz under Windows XP using 256 MB of RAM. Regarding the comparator algorithms described in McMullen (2001a), the relevant results for the same problem sets (Appendix B) were obtained directly from the author.

5.5. Comparison method

To evaluate performance of the MOGA approach, two measures were employed: Quality (Q) and Diversity (D). These are common measures for comparison of multi-objective optimization methods, for instance in Hyun et al. (1998) and Mansouri et al. (2003). Regarding small problems, i.e. problem sets 1 and 2, MOGA was compared against true non-dominated frontiers obtained via the TE scheme. The MOGA was also compared against the comparator algorithms in all problem sets. Quality of the locally non-dominated frontier found by an algorithm A when compared against algorithm B is calculated this way: locally non-dominated solutions of algorithm A (B) are mutually compared with those of the algorithm B (A). Ignoring dominated solutions, relative quality of A (B) when compared against B (A) is equal to the number of its remaining locally non-dominated solutions divided by the total number of solutions in its locally non-dominated frontier. Diversity is simply measured by the number of disjoint solutions in the locally non-dominated frontier.

Each problem was solved 20 times by the MOGA and the average figures as well as standard deviations obtained in these runs were reported.

5.6. Comparative results

Tables 1–5 present the averages (AVRG) and standard deviations (STDV) obtained in the 20 conducted runs for each of the test problems.

To examine the mean difference between the measures of the MOGA and those of the comparator algorithms and true non-dominated frontiers, pair-wise comparisons were made at significance level $\alpha = 0.01$ employing one-tail t-test (Bowker and Liberman, 1972). It was observed that with 99% confidence ($\alpha = 0.01$), the MOGA's quality is only 2% less than that of the TE scheme while diversity of the both are exactly the same in all replications. However, the average CPU time for the MOGA for problem sets 1 and 2 were 0.09 and 0.56 minutes, respectively. The same figures for TE were respectively 1.34 and 85.50 minutes. This justifies application of the MOGA even for small problems.

Table 1
Comparison of Quality (*Q*) and Diversity (*D*) in problem set 1

Algorithms		Problem sets																	
		1B		1C		1D		1E		1F		1G		1H		1I		1J	
		Q	*D*	*Q*	*D*	*Q*	*D*	*Q*	*D*	*Q*	*D*	*Q*	*D*	*Q*	*D*	*Q*	*D*	*Q*	*D*
TE		1.00	5	1.00	6	1.00	8	1.00	6	1.00	8	1.00	7	1.00	8	1.00	8	1.00	8
MOGA	AVRG	1.00	5	1.00	6	1.00	8	1.00	6	0.89	8	0.92	7	0.81	8	0.99	8	0.96	8
	(STDV)	(0.00)	(0.0)	(0.00)	(0.0)	(0.00)	(0.0)	(0.00)	(0.0)	(0.04)	(0.0)	(0.09)	(0.0)	(0.10)	(0.0)	(0.04)	(0.0)	(0.06)	(0.0)
GA	AVRG	1.00	5	1.00	6	1.00	8	1.00	6	0.75	8	1.00	7	0.03	8	0.75	8	1.00	8
	(STDV)	(0.00)	(0.0)	(0.00)	(0.0)	(0.00)	(0.0)	(0.00)	(0.0)	(0.00)	(0.0)	(0.00)	(0.0)	(0.06)	(0.0)	(0.00)	(0.0)	(0.00)	(0.0)
MOGA	AVRG	1.00	5	1.00	6	1.00	8	1.00	6	1.00	8	0.92	7	0.97	8	0.99	8	0.96	8
	(STDV)	(0.00)	(0.0)	(0.00)	(0.0)	(0.00)	(0.0)	(0.00)	(0.0)	(0.00)	(0.0)	(0.09)	(0.0)	(0.06)	(0.0)	(0.04)	(0.0)	(0.06)	(0.0)
SA	AVRG	1.00	5	1.00	6	1.00	8	1.00	6	0.88	8	0.71	7	0.25	8	0.76	8	1.00	8
	(STDV)	(0.00)	(0.0)	(0.00)	(0.0)	(0.00)	(0.0)	(0.00)	(0.0)	(0.00)	(0.0)	(0.00)	(0.0)	(0.00)	(0.0)	(0.03)	(0.0)	(0.00)	(0.0)
MOGA	AVRG	1.00	5	1.00	6	1.00	8	1.00	6	1.00	8	0.92	7	0.75	8	0.99	8	0.96	8
	(STDV)	(0.00)	(0.0)	(0.00)	(0.0)	(0.00)	(0.0)	(0.00)	(0.0)	(0.00)	(0.0)	(0.09)	(0.0)	(0.00)	(0.0)	(0.03)	(0.0)	(0.06)	(0.0)
TS	AVRG	1.00	5	1.00	6	1.00	8	1.00	6	0.75	8	1.00	7	0.33	8	0.88	8	1.00	8
	(STDV)	(0.00)	(0.0)	(0.00)	(0.0)	(0.00)	(0.0)	(0.00)	(0.0)	(0.00)	(0.0)	(0.00)	(0.0)	(0.06)	(0.0)	(0.00)	(0.0)	(0.00)	(0.0)
MOGA	AVRG	1.00	5	1.00	6	1.00	8	1.00	6	1.00	8	0.92	7	0.79	8	0.99	8	0.96	8
	(STDV)	(0.00)	(0.0)	(0.00)	(0.0)	(0.00)	(0.0)	(0.00)	(0.0)	(0.00)	(0.0)	(0.09)	(0.0)	(0.06)	(0.0)	(0.04)	(0.0)	(0.06)	(0.0)

Table 2
Comparison nf Quality (Q) and Diversity (T) in problem set 2

Algorithms		Problem sets																	
		2B		2C		2D		2E		2F		2G		2H		2I		2J	
		Q	D	Q	D	Q	D	Q	D	Q	D	Q	D	Q	D	Q	D	Q	D
TE		1.00	5	1.00	7	1.00	9	1.00	11	1.00	11	1.00	11	1.00	11	1.00	11	1.00	9
MOGA	AVRG	1.00	5	1.00	7	0.97	9	0.93	11	0.89	11	0.85	11	0.89	11	0.97	11	0.95	9
	(STDV)	(0.00)	(0.0)	(0.00)	(0.0)	(0.05)	(0.0)	(0.08)	(0.0)	(0.08)	(0.0)	(0.08)	(0.0)	(0.08)	(0.0)	(0.05)	(0.0)	(0.08)	(0.0)
GA	AVRG	1.00	5	1.00	7	1.00	9	0.36	11	0.00	11	0.00	11	0.18	11	0.36	11	1.00	9
	(STDV)	(0.00)	(0.0)	(0.00)	(0.0)	(0.00)	(0.0)	(0.00)	(0.0)	(0.00)	(0.0)	(0.00)	(0.0)	(0.00)	(0.0)	(0.00)	(0.0)	(0.00)	(0.0)
MOGA	AVRG	1.00	5	1.00	7	0.97	9	0.98	11	1.00	11	1.00	11	0.91	11	1.00	11	0.95	9
	(STDV)	(0.00)	(0.0)	(0.00)	(0.0)	(0.05)	(0.0)	(0.04)	(0.0)	(0.00)	(0.0)	(0.00)	(0.0)	(0.00)	(0.0)	(0.00)	(0.0)	(0.08)	(0.0)
SA	AVRG	1.00	5	1.00	7	1.00	9	1.00	11	0.50	11	0.82	11	0.64	11	0.28	11	1.00	9
	(STDV)	(0.00)	(0.0)	(0.00)	(0.0)	(0.00)	(0.0)	(0.00)	(0.0)	(0.05)	(0.0)	(0.00)	(0.0)	(0.00)	(0.0)	(0.02)	(0.0)	(0.00)	(0.0)
MOGA	AVRG	1.00	5	1.00	7	0.97	9	0.93	11	0.95	11	0.92	11	0.96	11	1.00	11	0.95	9
	(STDV)	(0.00)	(0.0)	(0.00)	(0.0)	(0.05)	(0.0)	(0.08)	(0.0)	(0.05)	(0.0)	(0.06)	(0.0)	(0.05)	(0.0)	(0.02)	(0.0)	(0.08)	(0.0)
TS	AVRG	1.00	5	1.00	7	1.00	9	1.00	11	0.71	10	0.27	11	0.70	10	0.10	10	1.00	9
	(STDV)	(0.00)	(0.0)	(0.00)	(0.0)	(0.00)	(0.0)	(0.00)	(0.0)	(0.03)	(0.0)	(0.00)	(0.0)	(0.00)	(0.0)	(0.00)	(0.0)	(0.00)	(0.0)
MOGA	AVRG	1.00	5	1.00	7	0.97	9	0.93	11	0.89	11	0.96	11	0.89	11	1.00	11	0.95	9
	(STDV)	(0.00)	(0.0)	(0.00)	(0.0)	(0.05)	(0.0)	(0.08)	(0.0)	(0.08)	(0.0)	(0.05)	(0.0)	(0.08)	(0.0)	(0.00)	(0.0)	(0.08)	(0.0)

Table 3
Comparison of Quality (Q) and Diversity (D) in problem set 3

Algorithms		Problem sets																	
		3B		3C		3D		3E		3F		3G		3H		3I		3J	
		Q	D	Q	D	Q	D	Q	D	Q	D	Q	D	Q	D	Q	D	Q	D
GA	AVRG	1.00	5	0.43	7	0.29	11	0.00	16	0.07	16	0.06	16	0.00	15	0.06	16	0.25	16
	(STDV)	(0.00)	(0.0)	(0.00)	(0.0)	(0.04)	(0.0)	(0.00)	(0.0)	(0.01)	(0.0)	(0.00)	(0.0)	(0.00)	(0.0)	(0.00)	(0.0)	(0.00)	(0.0)
MOGA	AVRG	0.20	5	1.00	7	0.98	11	1.00	14	1.00	16	0.98	16	1.00	15	0.94	16	0.85	13
	(STDV)	(0.00)	(0.0)	(0.00)	(0.0)	(0.05)	(0.0)	(0.00)	(0.0)	(0.01)	(0.0)	(0.03)	(0.0)	(0.00)	(0.0)	(0.00)	(0.0)	(0.00)	(0.0)
SA	AVRG	0.83	6	0.57	7	0.84	10	0.46	14	0.65	14	0.64	14	0.30	14	0.20	16	0.62	13
	(STDV)	(0.00)	(0.0)	(0.00)	(0.0)	(0.05)	(0.0)	(0.04)	(0.0)	(0.05)	(0.0)	(0.04)	(0.0)	(0.03)	(0.0)	(0.03)	(0.0)	(0.00)	(0.0)
MOGA	AVRG	0.20	5	1.00	7	0.93	11	0.90	14	0.95	16	0.93	16	0.98	15	0.98	16	0.99	13
	(STDV)	(0.00)	(0.0)	(0.00)	(0.0)	(0.09)	(0.0)	(0.04)	(0.0)	(0.06)	(0.0)	(0.06)	(0.0)	(0.04)	(0.0)	(0.04)	(0.0)	(0.03)	(0.0)
TS	AVRG	1.00	5	1.00	7	0.70	10	0.31	15	0.52	14	0.18	14	0.20	15	0.22	14	0.41	15
	(STDV)	(0.00)	(0.0)	(0.00)	(0.0)	(0.00)	(0.0)	(0.05)	(0.0)	(0.03)	(0.0)	(0.05)	(0.0)	(0.01)	(0.0)	(0.02)	(0.0)	(0.02)	(0.0)
MOGA	AVRG	0.20	5	1.00	7	0.96	11	0.92	14	0.91	16	0.97	16	0.98	15	0.98	16	1.00	13
	(STDV)	(0.00)	(0.0)	(0.00)	(0.0)	(0.05)	(0.0)	(0.05)	(0.0)	(0.06)	(0.0)	(0.03)	(0.0)	(0.03)	(0.0)	(0.04)	(0.0)	(0.00)	(0.0)

Table 4
Comparison of Quality (Q) and Diversity (D) in problem set 4

Algorithms		Problem sets																	
		4B		4C		4D		4E		4F		4G		4H		4I		4J	
		Q	D	Q	D	Q	D	Q	D	Q	D	Q	D	Q	D	Q	D	Q	D
GA	AVRG	0.40	10	0.00	10	0.09	11	0.00	11	0.00	11	0.00	11	0.00	11	0.01	9	0.91	11
	(STDV)	(0.00)	(0.0)	(0.00)	(0.0)	(0.00)	(0.0)	(0.00)	(0.0)	(0.02)	(0.0)	(0.00)	(0.0)	(0.00)	(0.0)	(0.03)	(0.0)	(0.00)	(0.0)
MOGA	AVRG	1.00	6	1.00	11	0.91	11	1.00	11	1.00	11	1.00	11	1.00	11	0.99	11	1.00	10
	(STDV)	(0.00)	(0.0)	(0.00)	(0.0)	(0.00)	(0.0)	(0.00)	(0.0)	(0.00)	(0.0)	(0.00)	(0.0)	(0.00)	(0.0)	(0.03)	(0.0)	(0.00)	(0.0)
SA	AVRG	0.60	10	0.06	10	0.23	10	0.08	10	0.24	9	0.17	9	0.15	10	0.01	11	0.00	11
	(STDV)	(0.00)	(0.0)	(0.07)	(0.0)	(0.06)	(0.0)	(0.10)	(0.0)	(0.05)	(0.0)	(0.08)	(0.0)	(0.12)	(0.0)	(0.03)	(0.0)	(0.00)	(0.0)
MOGA	AVRG	0.33	6	0.95	11	0.79	11	0.93	11	0.89	11	0.86	11	0.87	11	0.99	11	1.00	10
	(STDV)	(0.00)	(0.0)	(0.06)	(0.0)	(0.05)	(0.0)	(0.09)	(0.0)	(0.04)	(0.0)	(0.06)	(0.0)	(0.11)	(0.0)	(0.03)	(0.0)	(0.00)	(0.0)
TS	AVRG	0.90	10	0.21	10	0.41	10	0.28	10	0.67	9	0.32	10	0.27	10	0.16	9	0.45	11
	(STDV)	(0.00)	(0.0)	(0.02)	(0.0)	(0.03)	(0.0)	(0.09)	(0.0)	(0.00)	(0.0)	(0.05)	(0.0)	(0.09)	(0.0)	(0.07)	(0.0)	(0.00)	(0.0)
MOGA	AVRG	0.17	6	0.81	11	0.63	11	0.76	11	0.53	11	0.71	11	0.76	11	0.87	11	0.90	10
	(STDV)	(0.00)	(0.0)	(0.02)	(0.0)	(0.03)	(0.0)	(0.05)	(0.0)	(0.03)	(0.0)	(0.04)	(0.0)	(0.08)	(0.0)	(0.05)	(0.0)	(0.00)	(0.0)

Table 5
Comparison of Quality (Q) and Diversity (D) in problem set 5

Algorithms		Problem sets																	
		5B		5C		5D		5E		5F		5G		5H		5I		5J	
		Q	D	Q	D	Q	D	Q	D	Q	D	Q	D	Q	D	Q	D	Q	D
GA	AVRG	0.00	67	0.00	68	0.00	64	0.00	63	0.00	61	0.00	67	0.00	60	0.0	62	0.00	49
	(STDV)	(0.00)	(0.0)	(0.00)	(0.0)	(0.00)	(0.0)	(0.00)	(0.0)	(0.00)	(0.0)	(0.00)	(0.0)	(0.00)	(0.0)	(0.00)	(0.0)	(0.00)	(0.0)
MOGA	AVRG	1.00	53	1.00	57	1.00	59	1.00	62	1.00	65	1.00	65	1.00	64	1.00	63	1.00	61
	(STDV)	(0.00)	(2.7)	(0.00)	(2.6)	(0.00)	(3.5)	(0.00)	(3.2)	(0.00)	(3.0)	(0.00)	(2.7)	(0.00)	(2.8)	(0.00)	(2.2)	(0.00)	(2.6)
SA	AVRG	0.03	66	0.03	63	0.12	62	0.10	66	0.15	65	0.23	63	0.45	57	0.14	60	0.31	55
	(STDV)	(0.04)	(0.0)	(0.03)	(0.0)	(0.07)	(0.0)	(0.08)	(0.0)	(0.07)	(0.0)	(0.10)	(0.0)	(0.14)	(0.0)	(0.09)	(0.0)	(0.10)	(0.0)
MOGA	AVRG	0.98	53	0.97	57	0.94	59	0.93	62	0.87	65	0.81	65	0.66	64	0.90	63	0.73	61
	(STDV)	(0.02)	(2.7)	(0.03)	(2.6)	(0.05)	(3.5)	(0.05)	(3.2)	(0.06)	(3.0)	(0.09)	(2.7)	(0.10)	(2.8)	(0.07)	(2.2)	(0.08)	(2.6)
TS	AVRG	0.00	65	0.00	62	0.02	66	0.03	67	0.00	60	0.02	62	0.04	61	0.07	61	0.01	55
	(STDV)	(0.00)	(0.0)	(0.00)	(0.0)	(0.02)	(0.0)	(0.03)	(0.0)	(0.00)	(0.0)	(0.03)	(0.0)	(0.03)	(0.0)	(0.06)	(0.0)	(0.02)	(0.0)
MOGA	AVRG	1.00	53	1.00	57	0.98	59	0.97	62	1.00	65	0.99	65	0.97	64	0.94	63	0.99	61
	(STDV)	(0.00)	(2.7)	(0.00)	(2.6)	(0.02)	(3.5)	(0.03)	(3.2)	(0.00)	(3.0)	(0.02)	(2.7)	(0.03)	(2.8)	(0.06)	(2.2)	(0.02)	(2.6)

It was also observed that with 99% confidence, quality of MOGA is superior to that of GA, SA and TS by 123.6%, 47.8% and 41.3% respectively. In addition, with 99% confidence, no significant difference was observed between MOGA and the three comparator heuristics in terms of diversity. Moreover MOGA,

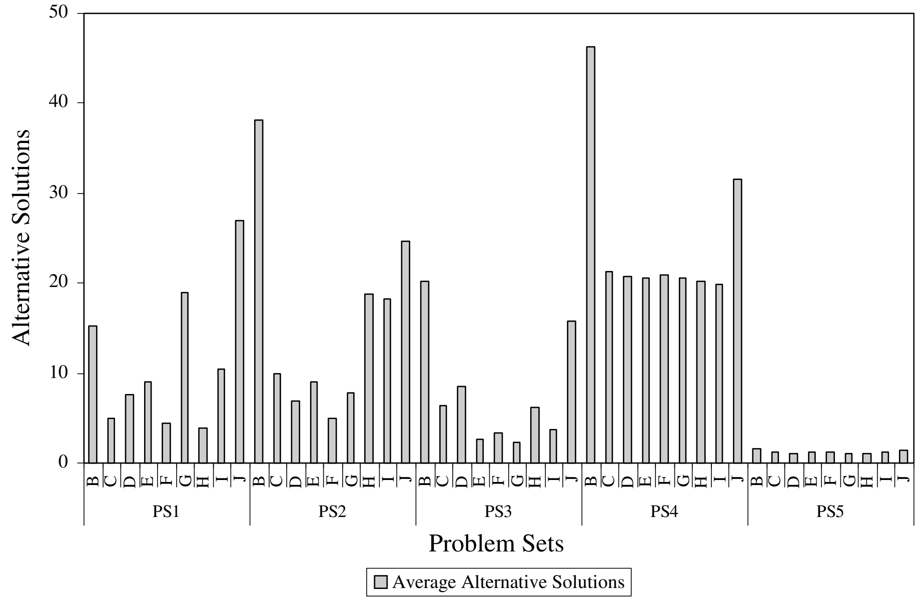

Fig. 3. Average number of alternative solutions obtained by MOGA.

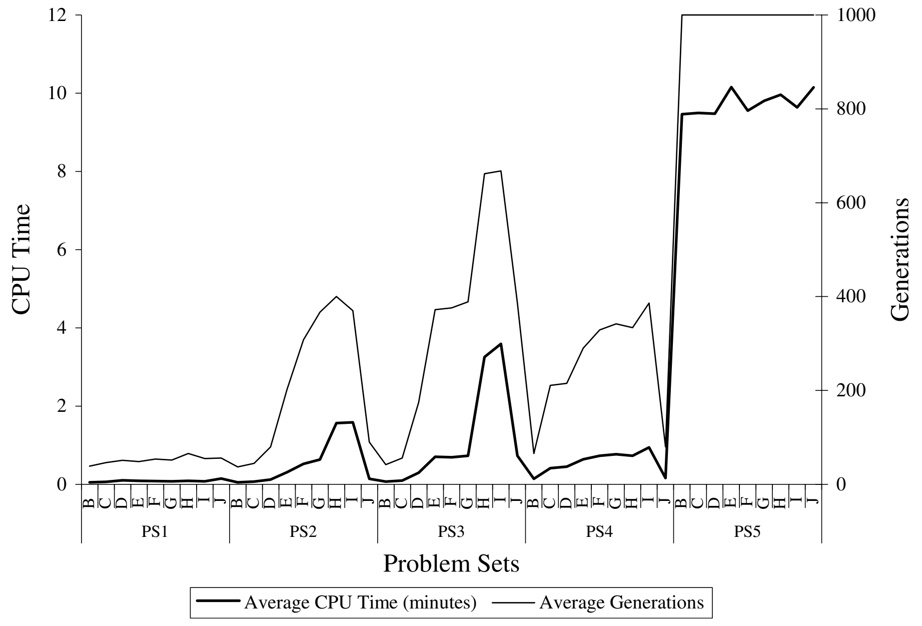

Fig. 4. CPU time and number of generations for the MOGA.

unlike its comparators, has produced several alternative sequences for each point along the locally non-dominated frontiers. This provides the decision-makers with more flexibility to decide on the desired sequence. Fig. 3 illustrates the average number of alternative solutions obtained by the MOGA for each disjoint solution along the locally non-dominated frontiers of the test problems.

The average CPU time for the MOGA as well as the average number of generations executed in problem sets are jointly plotted in Fig. 4. As it can be seen, the CPU requirement of the MOGA is reasonably low. The average CPU time for problem sets 1–5 were: 0.09, 0.56, 1.13, 0.55 and 9.74 minutes, respectively. The average generations for the same problems were: 52, 211, 347, 250 and 1000. Incidentally, the figure reflects a harmony between the number of generations and the amount of CPU time across the problems.

6. Conclusions

In this paper, a MOGA approach was introduced searching for locally Pareto-optimal frontier of a mixed-model sequencing problem in JIT environment where simultaneous minimization of setups and production rates variation is desired. These two objectives are conversely related to each other. Another attribute of the problem addressed in this paper is its combinatorial nature that makes traditional optimization techniques impractical for large problems.

The developed MOGA uses three basic genetic operators as crossover, inversion and mutation. It also exploits non-dominated sorting idea along with a niche mechanism to obtain quality as well as diverse locally Pareto-optimal solutions. An elitist strategy was also employed to preserve locally non-dominated frontiers found over generations from getting lost. A novel scheme for selection was used that selects individuals from the population as well as the elite set.

Extensive computational experiments were performed for the proposed algorithm. Performance of the MOGA was compared against a TE scheme in small problems. It was shown that the MOGA performs comparable with the TE while requiring considerably shorter CPU time. Performance comparisons of the MOGA against three existing algorithms for the same problem were also made for various test problems in terms of quality and diversity of the solutions. The results reveal that the proposed MOGA outperforms the benchmark algorithms in terms of the quality of solutions. Concerning diversity, no significant difference was observed between the MOGA and the benchmark algorithms, whereas the MOGA generates alternative solutions for each point along the locally non-dominated frontier.

It was observed that the niche mechanism using reduced dimensions has a significant positive impact on both quality and diversity at the same time. It was also observed that using the high rates for crossover and mutation would increase diversity whereas the low rates would improve quality. The both advantages were exploited using a changing rate using constant, high rates at the beginning followed by exponentially decreasing rates from a break point onward.

Besides the two objectives considered in this research, there are other criteria important to the production systems utilizing mixed-model lines. For instance: minimizing overall line length, utility work, risk of line stop, etc. Adjusting the proposed MOGA for mixed-model sequencing problems with other set of objectives provide avenues for further research.

Acknowledgments

The author wishes to express his gratitude to Dr. Patrick R. McMullen, Wake Forest University, USA for providing results of the benchmark algorithms and also to the referees and guest editors for their valuable comments on earlier draft of the paper.

Appendix A. Pseudocode for the selection scheme *RSSWR_UE*

```
For i = 1 to Population_Size
  Calculate Selection_Prob ($chromosome_i$)
  Calculate Expected_Number ($chromosome_i$)
  For j = 0 to [Expected_Number($chromosome_i$)]
      Copy $chromosome_i$ to Mating_Pool
  Next j
Next i
/* Initialize the Elite_Set */
For k = 1 to Elite_Set_Size
   Let Transfer_Prob($chromosome_k$) = Initial_Transfer_Prob
Next k
Do /* For current Population */
 For i = 1 to Population_Size
   If Rand<(Expected_Number($chromosome_i$)-[Expected_Number($chromosome_i$)]) Then
         If Rand<(Elitism_Prob) Then
         /* Select a chromosome from Elite_Set */
            Do
               For j = 1 to Elite_Set_Size
                  If Rand<Transfer_Prob($chromosome_j$) Then
                     Copy $chromosome_j$ to Mating_Pool
                     For k = 1 to Elite_Set_Size
                        If ($chromosome_k \in$ Niche_Cubicle($chromosome_j$) Then
                           Let Transfer_Prob ($chromosome_k$) =
                               Degrading_Factor$\times$Transfer_Prob($chromosome_k$)
                        End If
                     Next k
                  End If
               Next j
            While (A $chromosome_j$ has not been selected)
         End If
   Else If Rand$\not<$Transfer_Prob($chromosome_j$) Then
         /* Select a chromosome from current Population */
         Copy $chromosome_i$ to Mating_Pool
   End If
 Next i
While (Mating_Pool has not been filled)
```

Appendix B. Problem sets used in the experiments

Note: The number in the body of each table (Tables B.1–B.5) denotes the demand for that particular product.

Table B.1
Problem set 1

Problem	Product type					Solutions
	1	2	3	4	5	
B	8	1	1	1	1	11,880
C	7	2	1	1	1	47,520
D	6	3	1	1	1	110,880
E	6	2	2	1	1	166,320
F	5	3	2	1	1	332,640
G	5	2	2	2	1	498,960
H	4	3	2	2	1	831,600
I	4	4	2	1	1	415,800
J	3	3	2	2	2	1,663,200

Table B.2
Problem set 2

Problem	Product type					Solutions
	1	2	3	4	5	
B	11	1	1	1	1	32,760
C	10	2	1	1	1	180,180
D	9	3	1	1	1	600,600
E	7	5	1	1	1	2,162,160
F	7	3	2	2	1	10,810,800
G	6	3	3	2	1	25,225,200
H	5	3	3	3	1	50,450,400
I	4	3	3	3	2	126,126,000
J	3	3	3	3	3	168,168,000

Table B.3
Problem set 3

Problem	Product type					Solutions
	1	2	3	4	5	
B	16	1	1	1	1	1.163×10^5
C	15	2	1	1	1	9.302×10^5
D	13	4	1	1	1	1.628×10^7
E	10	5	2	2	1	1.397×10^9
F	8	7	2	2	1	2.993×10^9
G	6	6	5	2	1	1.995×10^{10}
H	5	5	5	3	2	1.173×10^{11}
I	5	4	4	4	3	2.444×10^{11}
J	4	4	4	4	4	3.055×10^{11}

Table B.4
Problem set 4

Problem	Product type										Solutions
	1	2	3	4	5	6	7	8	9	10	
B	11	1	1	1	1	1	1	1	1	1	6.095×10^{10}
C	10	2	1	1	1	1	1	1	1	1	3.352×10^{11}
D	9	3	1	1	1	1	1	1	1	1	1.117×10^{12}
E	8	4	1	1	1	1	1	1	1	1	2.514×10^{12}
F	7	5	1	1	1	1	1	1	1	1	4.023×10^{12}
G	6	5	2	1	1	1	1	1	1	1	1.408×10^{13}
H	5	5	3	1	1	1	1	1	1	1	2.816×10^{13}
I	4	4	4	2	1	1	1	1	1	1	2.112×10^{15}
J	2	2	2	2	2	2	2	2	2	2	2.376×10^{15}

Table B.5
Problem set 5

Problem	Product type															Solutions
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
B	40	40	8	1	1	1	1	1	1	1	1	1	1	1	1	3.477×10^{57}
C	35	35	10	5	5	1	1	1	1	1	1	1	1	1	1	1.673×10^{67}
D	30	30	15	10	5	1	1	1	1	1	1	1	1	1	1	2.329×10^{72}
E	25	25	20	15	5	1	1	1	1	1	1	1	1	1	1	1.016×10^{75}
F	20	20	20	15	15	1	1	1	1	1	1	1	1	1	1	3.790×10^{78}
G	20	20	15	15	10	6	6	1	1	1	1	1	1	1	1	4.901×10^{84}
H	15	15	15	10	10	10	10	5	4	1	1	1	1	1	1	8.357×10^{91}
I	15	15	10	10	10	10	10	10	4	1	1	1	1	1	1	9.959×10^{92}
J	7	7	7	7	7	7	7	7	7	6	6	6	6	6	6	6.334×10^{103}

References

Bard, J.F., Shtub, A., Joshi, S.B., 1994. Sequencing mixed-model assembly lines to level parts usage and minimize line length. International Journal of Production Research 32, 2431–2454.

Bowker, A.H., Liberman, G.J., 1972. Engineering Statistics, second ed. Prentice-Hall, Englewood Cliffs, NJ.

Cheng, L., Ding, F., 1996. Sequencing mixed-model assembly lines to minimize the weighted variations in just-in-time production systems. IIE Transactions 28, 919–927.

Coello, C.A.C., Van Veldhuizen, D.A., Lamont, G.B., 2002. Evolutionary Algorithms for Solving Multi-Objective Problems. Kluwer Academic Publishers, New York.

Deb, K., 2001. Multi-Objective Optimization using Evolutionary Algorithms. John Wiley, Chichester, UK.

Goldberg, D.E., 1989. Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley, Reading, MA.

Goldberg, D.E., Lingle, R., 1985. Alleles, loci, and the TSP. In: Proceedings of the First International Conference on Genetic Algorithms, pp. 154–159.

Hyun, C.J., Kim, Y., Kim, Y.K., 1998. A genetic algorithm for multiple objective sequencing problems in mixed model assembly lines. Computers & Operations Research 25 (7/8), 675–690.

Kubiak, W., Sethi, S.P., 1991. A note on level schedules for mixed-model assembly lines in just-in-time production systems. Management Science 37, 121–122.

Kubiak, W., Steiner, G., Yeoman, J.S., 1997. Optimal level schedules for mixed-model multi-level just-in-time assembly systems. Annals of Operations Research 69, 241–259.

Mansouri, S.A., Moattar-Husseini, S.M., Zegordi, S.H., 2003. A genetic algorithm for multiple objective dealing with exceptional elements in cellular manufacturing. Production Planning & Control 14 (5), 437–446.

McMullen, P.R., 1998. JIT sequencing for mixed-model assembly lines with setups using tabu search. Production Planning & Control 9 (5), 504–510.

McMullen, P.R., 2001a. An efficient frontier approach to addressing JIT sequencing problems with setups via search heuristics. Computers & Industrial Engineering 41, 335–353.

McMullen, P.R., 2001b. A Kohonen self-organizing map approach to addressing a multiple objective, mixed-model JIT sequencing problem. International Journal of Production Economics 72, 59–71.

McMullen, P.R., 2001c. An ant colony optimization approach to addressing a JIT sequencing problem with multiple objectives. Artificial Intelligence in Engineering 15, 309–317.

McMullen, P.R., Frazier, G.V., 2000. A simulated annealing approach to mixed-model sequencing with multiple objectives on a JIT line. IIE Transactions 32 (8), 679–686.

McMullen, P.R., Tarasewich, P., Frazier, G.V., 2000. Using genetic algorithms to solve the multi-product JIT sequencing problem with setups. International Journal of Production Research 38 (12), 2653–2670.

Michalewicz, Z., 1996. Genetic Algorithms + Data Structures = Evolution Programs. Springer, New York.

Miltenburg, J., 1989. Level schedules for mixed-model assembly lines in just-in-time production systems. Management Science 35 (2), 192–207.

Miltenburg, J., Goldstein, G.T., 1991. Developing production scheduling which balance part usage and smooth production loads in just-in-time production systems. Naval Research Logistics 38, 893–910.

Miltenburg, J., Steiner, G., Yeomans, S., 1990. A dynamic programming algorithm for scheduling mixed-model just-in-time production systems. Mathematical Computation Modeling 13, 57–66.

Monden, Y., 1993. Toyota Production System, second ed. The Institute of Industrial Engineers, Norcross, GA.

Oliver, I.M., Smith, D.J., Holland, J.R.C., 1987. A study of permutation crossover operators on the traveling salesman problem. In: Proceedings of the Second International Conference on Genetic Algorithms, pp. 224–230.

Srinivas, N., Deb, K., 1994. Multiobjective optimization using nondominated sorting in genetic algorithms. Evolutionary Computation 2 (3), 221–248.

Steiner, G., Yeomans, S., 1993. Level schedules for mixed-model, just-in-time processes. Management Science 39, 728–735.

Szidarovsky, F., Gershon, M.E., Dukstein, L., 1986. Techniques for Multiobjective Decision Making in Systems Management. Elsevier, New York.

T'kindt, V., Billaut, J.-C., 2002. Multicriteria Scheduling: Theory, Models and Algorithms. Springer, Berlin.

Tsai, L., 1995. Mixed model sequencing to minimize utility work and the risk of conveyor stoppage. Management Science 41, 485–495.

Walpole, R.E., Myers, R.H., 1985. Probability and Statistics for Engineers and Scientists, third ed. Macmillan, New York.

Zhu, J., Ding, F.-Y., 2000. A transformed two stage method for reducing the part-usage variation of the product-level and part-level solutions in sequencing mixed-model assembly lines. European Journal of Operational Research 127 (1), 203–216.