



---

# A problem space genetic algorithm in multiobjective optimization

AYTEN TURKCAN and M. SELIM AKTURK

*Department of Industrial Engineering, Bilkent University, 06533 Bilkent, Ankara, Turkey*

Received June 2001 and accepted December 2001

---

In this study, a problem space genetic algorithm (PSGA) is used to solve bicriteria tool management and scheduling problems simultaneously in flexible manufacturing systems. The PSGA is used to generate approximately efficient solutions minimizing both the manufacturing cost and total weighted tardiness. This is the first implementation of PSGA to solve a multiobjective optimization problem (MOP). In multiobjective search, the key issues are guiding the search towards the global Pareto-optimal set and maintaining diversity. A new fitness assignment method, which is used in PSGA, is proposed to find a well-diversified, uniformly distributed set of solutions that are close to the global Pareto set. The proposed fitness assignment method is a combination of a nondominated sorting based method which is most commonly used in multiobjective optimization literature and aggregation of objectives method which is popular in the operations research literature. The quality of the Pareto-optimal set is evaluated by using the performance measures developed for multiobjective optimization problems.

*Keywords:* Bicriteria scheduling, nonidentical parallel CNC machines, flexible manufacturing systems, local search, genetic algorithm, Pareto-optimality

## 1. Introduction

Many real world problems require simultaneous optimization of multiple objectives. In scheduling problems, several objectives are considered in the existing studies. The criteria based on due dates such as maximum tardiness, number of tardy jobs and total tardiness are related with the customer's satisfaction. The criteria based on completion times, utilization and inventory levels are mostly important for the manufacturer. Since most the criteria conflict with each other, a schedule optimizing one criterion may perform poorly for another criterion. A multicriteria approach can satisfy both the manufacturer's and customer's requirements. However, it is difficult to find a single optimum solution when the objectives conflict with each other. In this case, there are multiple efficient solutions which we cannot say that one solution is better than another one without knowing the decision maker's preferences. In this study, a bicriteria tool management and scheduling

problem is solved with the objectives of minimizing manufacturing cost that is of manufacturer's concern and total weighted tardiness that is of customer's concern. A problem space genetic algorithm with a new fitness assignment method is used to solve this multiobjective optimization problem. The algorithm finds "good" approximations to the efficient solution sets.

Before going through problem characteristics and the proposed algorithm, the basic concepts of multi-objective optimization will be explained in this section. A multiobjective optimization problem (MOP) can be formulated mathematically as follows:

$$\begin{aligned} \min f(x) &= (f_1(x), f_2(x), \dots, f_n(x)) \\ \text{st } x &\in X \end{aligned}$$

where  $x$  is a vector of discrete decision variables and  $X$  is a set of feasible solutions. When the objectives conflict with each other, a number of solutions known as Pareto-optimal or efficient solutions are found. A

solution  $x$  is said to *dominate*  $x'$  if  $f_i(x) \leq f_i(x')$  for all  $i \in \{1, 2, \dots, n\}$  and  $f_i(x) < f_i(x')$  for at least one  $i$ . If there is no solution like  $x$  which dominates  $x'$ , we can say that solution  $x'$  is Pareto-optimal or efficient. In the objective space, the vector of objective functions for an efficient solution is said to be a nondominated solution. The set of all Pareto-optimal solutions is called the Pareto-optimal set. The corresponding objective value vectors are called the Pareto-optimal front.

There are different ways for generating the Pareto-optimal set. One method is aggregating the objectives into a single objective function by using the weighted linear function. In this method, the objective function to be minimized is calculated as  $f(x) = w_1 f_1(x) + w_2 f_2(x) + \dots + w_n f_n(x)$ . The weights of the objectives are changed systematically to find approximately efficient solutions. The algorithms that are used for single objective optimization problems can also be used for solving multiobjective optimization problems when the objectives are aggregated into a single objective function, but several runs should be taken with different weight alternatives. The second method is the constraint method that takes  $n - 1$  of  $n$  objectives into the constraint set. The model is as follows:  $\{\min f(x) = f_j(x) | x \in X \text{ and } f_i(x) \leq b_i \text{ for } 1 \leq i \leq n, i \neq j\}$ , where  $b_i$  is the upper bound for objective  $i$ . The efficient solutions are found by changing upper bounds. One disadvantage of this method is that we should have an a priori knowledge about the range of upper bounds. Another method is goal programming. The objective function to be minimized in this method is  $f(x) = [\sum_{i=1}^n |f_i(x) - g_i|^r]^{1/r}$ , where  $g_i$  is the goal that should be achieved by objective  $i$ . The goals should be selected carefully such that they are close to the global Pareto-optimal set. The last method is the minmax which tries to minimize the worst objective value calculated as  $f(x) = \max\{w_1 f_1(x), \dots, w_n f_n(x)\}$ . All of these classical methods require a priori knowledge about the problem.

Local search heuristics, such as tabu search, simulated annealing and genetic algorithms, can be used to generate approximately efficient solutions. The most popular local search method in multi-objective optimization literature is the genetic algorithm (GA), which deals with a population of solutions and could find several solutions in a single run. Since different solutions are generated during the

evolutionary process of a genetic algorithm, someone might think that this set of solutions might lead to a nondominated set for a MOP. Although it is a tempting thought, it rarely happens in a real application. Therefore, one has to be very careful how the required parameters of a local search algorithm should be selected. For example, it is important to find a well-diversified solution set, but also a set of good solutions that will lead to a global Pareto set. That means a rather simple parameter of a fitness assignment and selection becomes a challenging problem in a MOP environment. That is why in this study a new fitness assignment and selection method was proposed and employed in a problem space genetic algorithm (PSGA). Computational results reported in this study indicate that the proposed fitness assignment and selection method is significantly better than the other approaches utilized in the multiobjective optimization literature. This study is important to highlight some of the difficulties that someone could face when a local search method, specifically the PSGA, is utilized to solve a MOP, and a set of suggestions to alleviate some of these problems.

The remainder of this paper is organized as follows. Tool management and scheduling problems are defined and trade-offs between two objectives are explained in Section 2. The basic steps of PSGA and its base heuristic are given in Section 3. In Section 4, we discuss the local search parameters, which could affect the performance of PSGA, including the proposed fitness assignment method and a set of performance measures that can be used to evaluate the quality of the nondominated solution set. In Section 5, we present the results of a computational study, then provide the concluding remarks in the last section. The notation used throughout this paper can be found in the appendix.

## 2. Problem definition

In this study, a real world application of solving tool management and scheduling problems simultaneously in flexible manufacturing systems (FMS) is considered. The production environment is as follows. There are nonidentical parallel CNC machines with different tool magazine capacities, maximum available horse powers, operating costs, and tool loading, replacing and changing times. Each machine can process one

part at a time. The parts have multiple operations and there is a precedence relationship among the operations of each part. Since part loading and unloading times are longer than tool change times in the existing CNC technology, all the operations of a part should be processed on the same machine. An operation should be performed with the tool that has enough remaining life, because the operation cannot be interrupted for a tool change due to surface finish quality. The tool magazines are integrated with the machines. Therefore, the machine should be stopped for tool replacement. Only one tool can be replaced at a time which implies that tool changing times are additive. There is a central tool storage area where the unassigned tools are kept. The tools are transferred between the central storage and the CNC machines with a robotic manipulator.

The overall aim is to integrate process level decisions such as machining conditions selection, tool allocation and tool replacement with system level decisions such as scheduling. Most of the existing studies solve these problems independently without considering the interaction between them. One practical problem in machining is selecting the proper cutting conditions, such as cutting speed and feed rate, for a given operation. In order to determine optimum cutting speed and feed rate for a given machining operation, several factors should be considered such as machining and tooling costs, surface finish requirements, upper limits imposed by machine horse power, and cutting tool material. We can decrease machining time, and hence the machining cost, by increasing the cutting speed or feed rate, but this will increase the tooling cost because of high utilization of tools.

While solving the tool management problem, tool allocation and replacement decisions are important. Since the tool magazines have limited capacities and parts use different tools for each operation, the tool magazines cannot be loaded with all necessary tools. Tool changes are required to process the operations. In most of the existing studies, tool changes are assumed to be due to part mix, even though tools are changed due to tool wear ten times more often than due to part mix as reported by Gray *et al.* (1993). The tool lives are not constant as assumed in most of the existing studies and are directly related with the machining conditions. The tool lives affect the frequency of tool changes due to wear and hence the nonmachining time which is incurred for tool changing, loading and

replacing. The machining and nonmachining times, which are determined by the decisions at process level, are the primary inputs to the scheduling problem. Solving tool management and scheduling problems independently may lead to suboptimal or even infeasible solutions.

In this study, there are two objectives that are in conflict with each other. The first objective is minimizing the total manufacturing cost comprised of machining, tooling and nonmachining costs. Machining cost is incurred for the time spent to complete a metal cutting operation. The machining cost for a single operation  $i$  of part  $p$  using tool  $j$  on machine  $m$  is calculated as

$$C_{pijm}^m = t_{m_{pijm}} \cdot C_{o_m} = \frac{\pi \cdot D_{pi} \cdot L_{pi}}{12 \cdot v_{pijm} \cdot f_{pijm}} \cdot C_{o_m}$$

Tooling cost is related with tool usage rate, which is the ratio of the machining time to tool life, and cost of the tool. It is calculated as follows:

$$\begin{aligned} C_{pijm}^{tool} &= U_{pijm} C_{t_j} = \frac{t_{m_{pijm}}}{T_{pijm}} C_{t_j} \\ &= \frac{(\pi D_{pi} L_{pi}) / (12 v_{pijm} f_{pijm})}{C_j / (v_{pijm}^{\alpha_j} f_{pijm}^{\beta_j} d_{pi}^{\gamma_j})} C_{t_j} \\ &= \frac{\pi D_{pi} L_{pi} d_{pi}^{\gamma_j}}{12 C_j v_{pijm}^{(1-\alpha_j)} f_{pijm}^{(1-\beta_j)}} C_{t_j} \end{aligned}$$

The cutting speed and the feed rate are the only decision variables that determine the machining and tooling costs. These cost terms are also in conflict with each other. When we increase speed or feed rate, the machining cost decreases, but the tooling cost increases. The nonmachining cost is incurred for the nonmachining time spent to change, replace and load tools and calculated as  $C_{pijm}^{nm} = C_{o_m} \cdot t_{pijm}^{nm}$ . The nonmachining cost depends on the current status of the tool magazine. Tool changing time,  $t_{c_m}$ , occurs when the currently used tool by the machine is not appropriate for operation and the required tool is already stored in the tool magazine. Tool loading time,  $t_{l_m}$ , is added to the nonmachining time when the required tool is not in the tool magazine and a free slot exists. Tool replacement time,  $t_{r_m}$ , occurs when there is no free slot for the required tool. In this case, a tool from the tool magazine should be removed in order to load the required tool. When speed or feed rate increases, the nonmachining time increases due to an

increase in tool usage rates and more frequent tool changes.

The second objective is to minimize the total weighted tardiness. The machining and nonmachining times are important in determining the weighted tardiness of part  $p$  on machine  $m$  which is calculated as

$$tard_{pm} = w_p \cdot \max\{0, t_m^{now} + t_{pm}^m + t_{pm}^{nm} - DD_p\}$$

The nonmachining cost and the total weighted tardiness are dynamic. That means, they change as the state of the system changes. The current status of the tool magazine determines the nonmachining time which affects the nonmachining cost and the total weighted tardiness.

In the literature, few studies consider bicriteria tool management and scheduling problems simultaneously. Tiwari and Vidyarthi (2000) proposed a GA to solve machine loading problem in FMS with the objectives of minimizing the system unbalance and maximizing the throughput. The weighted linear combination of two objectives is taken to find a single optimum solution. Akturk and Ozkan (2001) solved the identical parallel machine scheduling problem with the objective of minimizing the sum of tooling, operational and tardiness costs. They assume that all cost terms have equal weight and find a single optimum solution. Bernardo and Lin (1994) considered the nonidentical parallel machine scheduling problem with the objectives of minimizing the total tardiness and setup costs. The setup times were incurred because of the material handling and tooling constraints. An interactive branch-and-bound algorithm, which allowed the decision maker to evaluate the solutions, was proposed to find efficient solutions. In a previous study by Turkcan *et al.* (2001), we solved these two problems simultaneously, but we did not deliberately work on finding approximately efficient solutions. We only stored the nondominated solutions that we have found during the search and selected one solution according to the aggregated value of the two objectives. In this study, we use the PSGA with a new fitness assignment method to find a ‘‘good’’ approximation to the efficient solution set.

### 3. The problem space genetic algorithm (PSGA)

The problem space search, which is proposed by Storer *et al.* (1992), is a local search method. It

provides a new neighborhood structure defined in the space of possible problem data perturbations. In problem space search, a fast, problem-specific ‘‘base heuristic’’ is required which maps a problem instance into a solution. The base heuristic finds several solutions for different perturbation vectors which are used to perturb the original problem data. For example, in single machine scheduling, the shortest processing time (SPT) rule can be selected as the base heuristic. The only relevant problem data that can be perturbed is the processing times of the jobs. The processing times are perturbed with randomly generated perturbations and the base heuristic, SPT, is used to schedule the jobs with the perturbed problem data. In problem space search, the objective function values of the solutions are evaluated by using the original problem data. In PSGA, GA is used to generate different perturbation vectors that are the encodings of each problem instance. We can find several solutions by using this method, but they may not lead to a global nondominated set. The main advantage of PSGA is that there is no need for a feasibility check, which requires a significant amount of computation time in our problem.

The PSGA is applied to single objective optimization problems up to now. The aim in single objective optimization problems is to find a single solution giving the minimum objective function value. Therefore, the convergence of PSGA is important. In MOP, the aim of genetic algorithms are guiding the search towards the global Pareto-optimal set and maintaining a diversified set of nondominated solutions. The parameters in PSGA that are suitable for single objective optimization may not be suitable for multiobjective optimization. In this section, we will first give the basic steps of PSGA and explain the base heuristic. The basic steps of PSGA are as follows:

*Step 1: (Initialization)* Set the generation number  $k$  equal to zero and form the initial perturbation matrix,  $A^k$ .

$$A^k = \begin{matrix} 1 \\ \vdots \\ P \end{matrix} \begin{bmatrix} \delta_{11}^k & \delta_{12}^k & \cdots & \delta_{1s}^k \\ \vdots & \vdots & & \vdots \\ \delta_{p1}^k & \delta_{p2}^k & \cdots & \delta_{ps}^k \end{bmatrix}$$

where the  $\delta_{ij}^k$  values are randomly generated from distribution  $UN \sim [-\theta, \theta]$  where  $UN \sim [a, b]$  stands for uniform distribution between  $a$  and  $b$ .

*Step 2:* For each perturbation vector,  $i \in \{1, 2, \dots, P\}$ , calculate the manufacturing cost,  $f_i^m$ , and total weighted tardiness,  $f_i^t$ , by calling the “base heuristic”.

*Step 3:* Update the nondominated solution set according to the solutions found in generation  $k$ . Increase  $k$  by 1. If  $k < \text{MAXGENS}$ , go to Step 4. Otherwise, STOP.

*Step 4:* (Fitness assignment) For each encoding  $i$ , calculate the fitness value,  $\text{fitness}(i)$ . Assign probabilities to each individual  $i$ ,  $\text{Prob}(i)$ , according to the fitness values.

*Step 5:* (Reproduction) Generate  $P$  encodings for generation  $k$  by using asexual and sexual reproduction. In asexual reproduction, an individual is selected and directly copied to the next generation. In sexual reproduction, two individuals are selected and mated to form a new individual. Elitism, where the “best” individuals are copied to the next generation, can also be used for generating perturbation vectors.

*Step 6:* (Mutation) Mutate an individual  $i$  with a predetermined mutation rate to form the perturbation matrix for generation  $k$ ,  $A^k$ . Go to Step 2 with the new perturbation matrix.

A fast, problem-specific base heuristic, which maps a problem instance to a solution, is very important for the efficiency of problem space search. In this study, we will use the base heuristic proposed by Turkcan *et al.* (2001). The basic steps of the base heuristic are as follows:

*Stage 1:* Find optimal machining conditions for each operation-tool pair.

*Stage 2:* Scheduling the nonidentical parallel CNC machines.

*Step 2.1:* Initialize the set of unscheduled jobs as  $\text{UNS} = \{1, 2, \dots, N\}$  and altered machines as  $\text{ALT} = \{1, 2, \dots, M\}$ .

*Step 2.2:* For each unscheduled part  $p$  and altered machine  $m$ , calculate the increase in both objective functions such that  $f_{pm}^m = \sum_i \sum_j (C_{pijm}^m + C_{pijm}^{\text{tool}} + C_{pijm}^{\text{nm}})$  and  $f_{pm}^t = w_p \max\{0, t_m^{\text{now}} + t_{pm}^m + t_{pm}^{\text{nm}} - DD_p\}$ .

*Step 2.3:* For each unscheduled part, find the machine giving the minimum weighted linear function of two normalized objectives, denoted by  $m_p$ .

*Step 2.4:* Calculate the following part selection index,  $\text{PI}_{pm_p}$ , for each  $\{p, m_p\}$  pair:

$$\text{PI}_{pm_p} = \frac{w_p}{t_{pm_p}^m + t_{pm_p}^{\text{nm}}} \times \exp \left[ - \frac{\max(0, DD_p - t_{m_p}^{\text{now}} - t_{pm_p}^m - t_{pm_p}^{\text{nm}})}{k \cdot \bar{t}} \right]$$

where  $\bar{t} = \sum_{p \in \text{UNS}} (t_{pm_p}^m + t_{pm_p}^{\text{nm}}) / |\text{UNS}|$  and  $k$  is a lookahead parameter. Select the part-machine pair giving the maximum  $\text{PI}_{pm_p}$ , say  $\{p^*, m_{p^*}\}$  pair.

*Step 2.5:* Assign part  $p^*$  to machine  $m_{p^*}$ . Update the current time on machine  $m_{p^*}$  such that  $t_{m_{p^*}}^{\text{now}} = t_{m_{p^*}}^{\text{now}} + t_{p^*m_{p^*}}^m + t_{p^*m_{p^*}}^{\text{nm}}$ . The remaining lives of tools used by part  $p^*$  are updated as  $R_{jm_{p^*}} = R_{jm_{p^*}} - U_{p^*ijm_{p^*}}$ .

*Step 2.6:* Update  $\text{UNS} = \text{UNS} \setminus \{p^*\}$  and  $\text{ALT} = \{m_{p^*}\}$ . If  $\text{UNS} \neq \emptyset$ , go to Step 2.2. Otherwise, STOP.

In the first step, optimal machining conditions are found by using a geometric programming model proposed by Akturk and Avci (1996). The model finds cutting speed and feed rate for each operation-tool pair by minimizing the sum of machining and tooling costs subject to tool life, machine power and surface roughness constraints. Since there are alternative tools for each operation, the tool alternative giving the minimum objective function value is selected to perform the corresponding operation. In the second step, parts and tools are scheduled on nonidentical parallel CNC machines. The parts are assigned to machines one at a time and the nonmachining times which affect the nonmachining costs and weighted tardiness are recalculated after each assignment, since they change according to the tool magazine status. While calculating the nonmachining time, a critical decision should be given if the required tool to perform an operation is not loaded on the tool magazine and there is no free slot for that tool. The tool that will be removed from the tool magazine to leave a free slot for the required tool should be selected. A tool removal index,  $\text{TI}_{jm}$ , which considers the remaining life of tool  $j$ ,  $R_{jm}$ , and the number of operations that can be performed by the remaining tool life,  $\sum x_{pijm}$ , is proposed to select the tool to be removed. The tool removal index is calculated as  $\text{TI}_{jm} = R_{jm} \sum x_{pijm}$  and the tool having minimum  $\text{TI}_{jm}$  is removed from the tool magazine. At each decision point, the machine which gives the minimum weighted average of two normalized objective function values is selected as the most appropriate machine for the corresponding part,  $m_p$ . This machine

can change at another time point, because the nonmachining costs and weighted tardiness values are dynamic and depend on the current tool magazine status. A part is selected for assignment according to a proposed part selection index,  $PI_{pm}$ . The proposed part selection index considers machining time, nonmachining time and the slack. It gives higher priority to the parts having less slack and shorter weighted processing time that is the sum of machining and nonmachining times. This procedure continues until all the parts are scheduled.

The base heuristic is used to find a schedule for each problem instance. The PSGA calls the base heuristic with different perturbation vectors to find several solutions. This is the first implementation of PSGA to a MOP in order to find the nondominated solution set. Since PSGA is used in single objective optimization problems up to now, some parameters that are used in previous studies may not be suitable for MOPs. The parameters that could affect the performance of PSGA in a MOP will be discussed in the next section.

#### 4. PSGA parameters

In single objective optimization problems, the parameters of a local search algorithm are selected such that the algorithm can quickly converge to the global optimum. In multiobjective optimization, the aim is changed as finding a well-diversified, evenly distributed set of solutions that are close to the global Pareto front. One could face some difficulties when a local search algorithm, designed to solve a single objective optimization problem, is used to solve a MOP. The first difficulty is about the convergence of local search methods. The GAs control the convergence with the fitness assignment and selection operations as discussed in Section 4.1. The other difficulties are the diversity and the quality of the approximated nondominated set as discussed in Sections 4.2 and 4.3, respectively.

##### 4.1. Fitness assignment and selection

In single objective optimization problems, the objective function value of a solution is used as the fitness value of that individual and the fittest individuals are selected for generating new individuals. The idea is that the solutions with better

objective function values will most probably lead us to the global optimum. In MOPs, the fitness assignment and selection becomes a challenging problem. This happens because it is difficult to decide which objective function(s) will be used to determine the fitness of an individual. Therefore, different approaches are developed for fitness assignment and selection. The first method is selection by switching objectives (Schaffer, 1985). At each decision point for reproduction, a different objective is used for evaluating the individuals. This method is useful to find a well-diversified solution set, but the convergence might be slower. The second method is aggregation selection with parameter variation (Hajela and Lin, 1992). The objectives are aggregated into a single objective function with different weights for each objective. The problem with this method is selecting the function that will be used for aggregating the objectives. Taking the weighted linear function of all objectives is the most popular method in operations research literature. In multiobjective optimization literature, using the weighted Tchebycheff function is suggested. Because, when the decision maker's preferences are unknown, the linear function of the objectives could not find all nondominated solutions. A diversified set of solutions is found by changing the relative importance of objectives systematically. This method does not use the nondominance relations between the solutions, although the aim is to find the nondominated solution set.

The last method is Pareto-ranking based selection (Zitzler, 1999; Srinivas and Deb, 1994). The fitness of an individual depends on its Pareto-dominance determined according to the whole population and/or the external nondominated set. Higher fitness values are assigned to the nondominated solutions. This method is the most popular one in multiobjective genetic algorithms (MOGA) literature, because it does not need any function for combining the objectives. However, without an objective function, it is more difficult to converge to the global Pareto-set, especially when the problem is difficult. In Pareto-ranking based methods, different methods are used to maintain diversity such as fitness sharing, restricted mating and re-initialization. In fitness sharing method, the fitness of an individual that has higher number of neighbors is less than another individual having fewer neighbors. In restricted mating method, two individuals are allowed to mate if they are within a certain distance. The idea is that mating the solutions that are

far from each other will most probably not lead to a good solution. The re-initialization is another method that initializes all the population when the search stagnates. A detailed literature review on MOGA can be found in Van Veldhuizen and Lamont (2000).

In this study, we proposed a new fitness assignment and selection method which is used in the fourth and fifth steps of PSGA. The proposed fitness assignment scheme is a combination of a Pareto-ranking based method and aggregation with parameter variation method. According to the proposed nondominated sorting and aggregation with parameter variation based (NSAPV) method, the following steps are performed to find the fitness of each individual  $i$ :

*Step 4.1:* Set  $P_{remain} = \{1, 2, \dots, P\}$  and dummy fitness =  $2 \cdot P$ .

*Step 4.2:* Determine the set of “relatively” nondominated solutions in  $P_{remain}$ , denoted by  $P_{nds}$ .

*Step 4.3:* Perform fitness sharing in objective space and only within the set  $P_{nds}$ .

$$\text{fitness}(i) = \frac{\text{dummy fitness}}{(1 + \sum_{j \in P_{nds}} s(d(i,j)))(1 + f_i^{agg})}$$

where

$$s(d(i,j)) = \begin{cases} 1 - \left(\frac{d(i,j)}{\sigma_{share}}\right)^2 & \text{if } d(i,j) \leq \sigma_{share} \\ 0 & \text{otherwise} \end{cases}$$

and  $f_i^{agg} = w \cdot f_i^{m'} + (1 - w) \cdot f_i^{f'}$ .

If individual  $i$  is not dominated by the external nondominated solution set, multiply  $\text{fitness}(i)$  by 2 in order to give higher priorities to the nondominated solutions with respect to the external set.

*Step 4.4:* Update dummy fitness such that  $\{0 < \text{dummy fitness} < \min_{i \in P_{nds}} \{\text{fitness}(i)\}\}$  and  $P_{remain} = \{P_{remain} \setminus P_{nds}\}$ .

*Step 4.5:* If  $P_{remain} = \emptyset$ , go to Step 4.6. Otherwise, go to Step 4.2.

*Step 4.6:* Calculate probability of selecting an individual  $i$  according to its fitness value as follows:

$$\text{Prob}(i) = \frac{\text{fitness}(i)}{\sum_{j \in P} \text{fitness}(j)}$$

The proposed fitness assignment method first finds the nondominated solutions in the population. This set is the closest trade-off front to the global Pareto-optimal front, which is front 1 in Fig. 1. The highest fitness

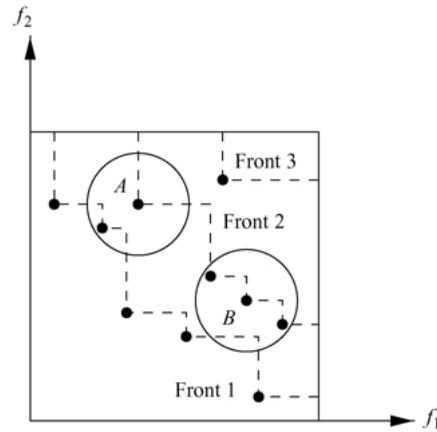


Fig. 1. Fitness sharing.

values are assigned to the solutions in the first front. Fitness sharing in objective space is used to find a uniformly distributed nondominated solution set. The solutions that have fewer neighbors in  $\sigma_{share}$ -neighborhood have higher fitness values. One important issue is that fitness sharing is performed among the solutions of same front. The solutions from different fronts are not counted as neighbors. If we count them, the dominated solutions are penalized twice and this will lead to premature convergence. For example, in Fig. 1, point A has one solution in its  $\sigma_{share}$ -neighborhood, but it is not counted as a neighbor since they are not on the same front. Point B has two neighbors, which are on the same front with B. The external nondominated set also affects the fitness assignments. The fitness values of the solutions, which cannot be dominated by the solutions of the external nondominated solution set, are increased. We incorporate the aggregated objective function value into our fitness assignment method. A solution that has smaller aggregated objective function value has a higher fitness value since it is closer to the global Pareto-set. This will help us guiding the search towards the global set. We use different weight alternatives for aggregating the objectives. After the fitness values are assigned to the individuals in the first front, these solutions are peeled off and the same procedure is repeated for the remaining solutions in the population.

We compare the proposed fitness assignment method with three other methods from the literature. In the first method, called Linear, the weighted linear

function of two normalized objectives are taken as the fitness of individual  $i$  such that  $\text{fitness}(i) = wf_i^{m'} + (1-w)f_i^{f'}$ . In the Tchebycheff method, the weighted Tchebycheff function of two normalized objectives is used for fitness assignment. The fitness of an individual is calculated as  $\text{fitness}(i) = \max\{wf_i^{m'}, (1-w)f_i^{f'}\}$ . In these two methods, the probability of selecting an individual  $i$  is calculated as  $\text{Prob}(i) = [\max_k \text{fitness}(k) - \text{fitness}(i)]^\pi / \sum_j [\max_k \text{fitness}(k) - \text{fitness}(j)]^\pi$ , where  $\pi$  is set to 2 in our experimental setting. As  $\pi$  increases, we can get a diversified set of solutions, but the convergence is slower. The third method is a nondominated sorting based method (NSGA) proposed by Srinivas and Deb (1994).

The fitness of an individual is calculated as  $\text{fitness}(i) = \text{dummy fitness} / \sum s(d(i,j))$ . We add 1 to the denominator, because when there is no solution in the neighborhood of a solution, the fitness goes to infinity and the probability of selecting that individual will be one. The selection mechanism fails, since other individuals in the population will have no chance to be selected. Furthermore, Srinivas and Deb (1994) use fitness sharing in decision space in their study, but we use fitness sharing in objective space. When we try to use fitness sharing in decision space, the perturbation vectors that are close to each other give almost the same schedules. This is very undesirable in the PSGA since we could not generate different solutions.

#### 4.2. Population diversity

The second problem one could face while using local search methods in MOPs is about the diversity of the nondominated solution sets. The proposed fitness assignment method uses fitness sharing in objective space in order to find a well-diversified and uniformly distributed set of nondominated solutions. We also incorporate the aggregated objective function value into the fitness assignment method. In the base heuristic, we aggregate the increase in both objective functions to find the most appropriate machine for each part at each decision point. Taking a fixed weight will lead the search to a certain region of the global Pareto-set. Therefore, different weight alternatives should be used to search different parts of the search space. The problem in the proposed method is how one should determine the weight alternatives. One method is to use a fixed weight throughout all

generations and repeating the same procedure for different weight alternatives such as 0.1, 0.3, 0.5, 0.7 and 0.9.

The other possibility is dynamic weighting method proposed by Jin *et al.* (2001). In this method, a number of weight alternatives changing throughout the generations are used for a number of times. The dynamic weighting method finds weights for each generation as  $w = |\sin 2\pi k/F|$ , where  $k$  is the generation number and  $F$  is the frequency which is used to determine how many times a weight alternative is used. For example, when the maximum number of generations is 15 and  $F$  is equal to 20, the weight,  $w$ , changes as  $\{0.0, 0.3090, 0.5878, 0.8090, 0.9511, 1.0, 0.9511, 0.8090, 0.5878, 0.3090, 0.0, |-0.3090|, |-0.5878|, |-0.8090|, |-0.9511|, |-1.0|\}$  for generations 0 through 15, respectively. In Jin *et al.* (2001), the weights change throughout the generations in the interval  $[0.0 \ 1.0]$ , but our initial runs indicated that the convergence to the global Pareto-set is very slow. Therefore, we decided to divide the region  $[0.0 \ 1.0]$  into five subregions of  $[0.0 \ 0.2]$ ,  $[0.2 \ 0.4]$ ,  $[0.4 \ 0.6]$ ,  $[0.6 \ 0.8]$  and  $[0.8 \ 1.0]$ . The weights are changed according to sinus function in each interval, and the procedure is repeated for each region. After some trial runs, we have seen that the dynamic weighting method gives better results than the fixed weight method. Therefore, we use dynamic weighting method in our experimental runs, and the frequency,  $F$ , is set to 40.

The other PSGA parameters such as the problem data that is perturbed, the perturbation values, the mutation probabilities, the percentage of sexual reproduction, the population size, the maximum number of generations and the number of restarts affect both the convergence to the global Pareto set and diversity of the nondominated solutions. The first parameter is the problem data that is perturbed. In the proposed base heuristic, we can perturb either the aggregated objective function value which is used for machine selection, or the part selection index which is used for selecting a part-machine pair, or the tool removal index which is used to remove a tool from the tool magazine when there is no empty slot for the required tool. In our trial runs, we see that the solution quality degrades when we perturb all of the three indices simultaneously. In this case, guiding the search towards the global Pareto-optimal set is more difficult. On the other hand, when we perturb only one problem data, we cannot maintain diversity.

Therefore, we decide to perturb both part and tool selection indexes at the same time. The second parameter is the value of perturbations which is taken randomly from the distribution  $UN \sim [-\theta, \theta]$ . When  $\theta$  is too small, we cannot generate a diversified set of solutions. When it is too large, the nondominated solution set will be far from the global Pareto set. After some trial runs,  $\theta$  is set to 0.5.

The reproduction and mutation operators aim at generating new perturbation vectors, hence lead to new solutions, by changing the existing ones. In order to find different solutions, the percentage of sexual reproduction, where two individuals are selected and mated to form a new individual, is taken as 80%. The mutation operator changes some parts of the encoding with a certain probability. Taking the mutation probability too small will lead to premature convergence of the GA. When it is large, the effect of fitness assignment and selection decreases, since randomness increases. It is more difficult to converge to the global Pareto-optimal set. The mutation probability is set to 0.05 after some trial runs.

The population size, maximum number of generations and number of restarts are other factors that can affect the performance of PSGA. As the population size increases, we can find a well diversified nondominated solution set. As the maximum number of generations increases, the Pareto-optimal set will be closer to the global Pareto-optimal set. The number of restarts also affects the diversity of the nondominated solutions. For single start runs, the convergence to the global set is better. On the other hand, for multiple start runs, diversity is better. We take four different levels having different population sizes, maximum number of generations and number of restarts. In the first level, population size is 20, maximum number of generations is 30 and number of restarts is 5, denoted by (20,30,5). The other levels are (20,150,1), (30,40,5) and (30,200,1).

Although several authors indicated that elitism could improve the performance of GAs in MOPs, few studies investigate the effect of elitism (Deb and Goel, 2000 and Zitzler, 1999). In single objective optimization problems, the strategy of copying the best individual of a generation to the next generation is known as elitism. In MOP, all nondominated solutions found up to that point are denoted as best solutions. If all nondominated solutions are passed to the next generation, the GA may stagnate after a certain point and prematurely converge to a suboptimal solution

set. In MOGAs, the problem is selecting solutions that should be passed to the next generation. In order to see the effect of using elitism on the performance of PSGA, we use two different elitism methods in our study. In the first method, we pass the solutions giving minimum manufacturing cost and total weighted tardiness to the next generation. In the second method, we increase the cardinality of the elite set by inserting some solutions from the external nondominated solution set. When the cardinality of the elite set is equal to three, a line is drawn between the two solutions giving the minimum manufacturing cost and total weighted tardiness. The solution from the nondominated set that is the closest one to the midpoint of that line is inserted to the elite set. When the cardinality of the elite set is increased to four, two nondominated solutions are selected for insertion. The distances between each member of the elite set are almost equal.

#### 4.3. Quality of a nondominated set

In multiobjective optimization problems, evaluating the quality of the approximations to the Pareto optimal set is important. The evaluations are useful for comparing different algorithms, defining stopping rules for algorithms and adjusting parameters of the algorithms. There are different performance measures that are used in the literature. The first performance measure is the area proposed by Zitzler and Thiele (1998). It is the size of the objective value space covered by a set of nondominated solutions. The second performance measure is the coverage difference of two sets,  $CD(A, B)$  (Zitzler, 1999). It is calculated as  $CD(A, B) = \text{area}(A \cup B) - \text{area}(B)$ . This measure is important in order to see the contribution of set A to the area covered by set B. These two measures use the ideal point information for calculating the areas. It is difficult to estimate the ideal point in complex problems. In this study, we use the best objective function values obtained from all algorithms as the ideal point. The third performance measure is the expected utility proposed by Hansen and Jaszkiewicz (1998) and calculated as  $EU(A) = \int_{u \in [0,1]} f(A(u)) du$ , where  $f(A(u)) = \min_{v \in A} \{uf^{ml}(x) + (1-u)f^t(x)\}$  or  $f(A(u)) = \min_{v \in A} \{\max\{uf^{ml}(x), (1-u)f^t(x)\}\}$ . With this performance measure, we try to find the expected aggregated objective function value where the expectation is taken over the relative importance of objectives. The advantage of this

method is that the performance of an algorithm can be evaluated independently of other algorithms. The next performance measure is the probability that an algorithm,  $A$ , gives a better solution than another algorithm,  $B$ ,  $R1(A, B)$ . It is also proposed by Hansen and Jaszkiewicz (1998) and calculated as  $R1(A, B) = \int_{u \in [0,1]} C(A(u), B(u)) du$ , where

$$C(A(u), B(u)) = \begin{cases} 1 & f(A(u)) < f(B(u)) \\ 1/2 & f(A(u)) = f(B(u)) \\ 0 & f(A(u)) > f(B(u)) \end{cases}$$

The last performance measure is the *coverage* (Zitzler and Thiele, 1998), which is used to derive whether a set entirely dominates the other. Coverage of set  $A$  over set  $B$ ,  $coverage(A, B)$ , is the fraction of solutions in set  $B$  which are covered by solutions in set  $A$  and calculated as follows:

$$coverage(A, B) = \frac{|\{z'' | \exists z' \in A : z' \preceq z''\}|}{|B|}$$

These five performance measures should not be interpreted alone, because each performance measure can give some additional information about the quality of a nondominated set. For example, let us take two hypothetical nondominated solution sets,  $A$  and  $B$ , of our problem as shown in Fig. 2. The area covered by algorithm  $A$  is 0.8557, and it is 0.859 for algorithm  $B$ . Since the difference is insignificant, we cannot decide which algorithm is better. However, when we look at the coverage measure, we can see that  $coverage(A, B)$  is 0.1609, while  $coverage(B, A)$  is 0.76, that means, 76% of the solutions in set  $A$  is

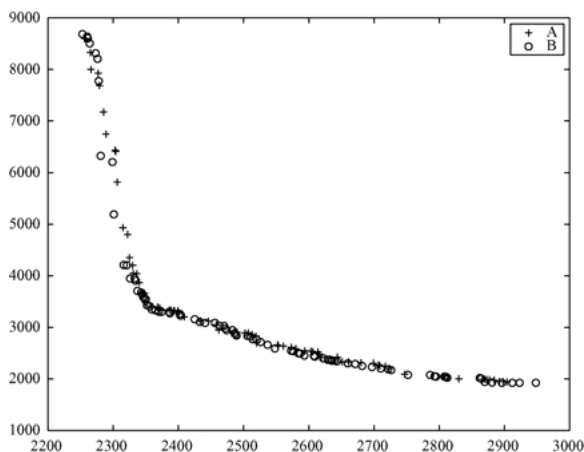


Fig. 2. Advantage of coverage and R1.

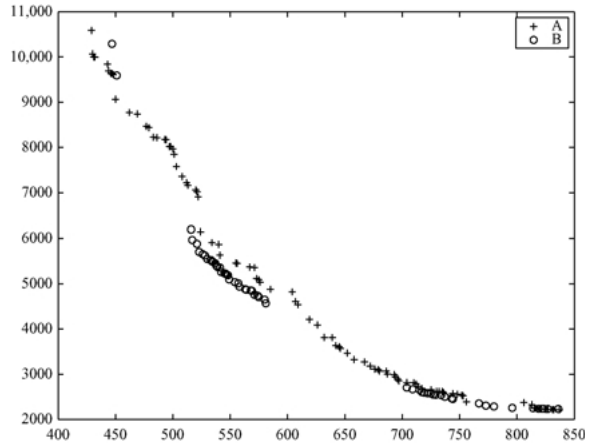


Fig. 3. Disadvantage of coverage.

covered by the solutions of set  $B$ . In addition, the probability that  $B$  gives a better solution than  $A$  is 0.785. These results show that  $B$  is significantly better than  $A$ , although we cannot get this information from the area measure. The coverage may not be a good measure either when it is interpreted alone. In Fig. 3,  $coverage(A, B)$  is 0.07, while  $coverage(B, A)$  is 0.371. However, as we can see from the figure, the nondominated solutions found by set  $B$  are not uniformly distributed over the entire region. In this case, we have to look at other performance measures. For example, the probability that  $A$  gives better solutions than  $B$  is 0.565 which shows that  $A$  is marginally better than  $B$ . The other measures also give the same result as  $R1$ .

In this study, we use these five performance measures for adjusting PSGA parameters, which were discussed above, and comparing the proposed method with the existing methods in literature, which will be discussed in the next section.

### 5. Computational analysis

We performed a computational study in order to test the performance of PSGA with the proposed fitness assignment method in a multiobjective optimization problem. We compared the proposed fitness assignment method with two classical approaches, aggregation of two objectives by using weighted linear or Tchebycheff distance functions, and a nondominated sorting based algorithm. The algorithms were coded in C language and compiled with

**Table 1.** Experimental design factors

| Factor | Definition           | Low level                          | High level                         |
|--------|----------------------|------------------------------------|------------------------------------|
| A      | Number of operations | UN ~ [2, 6]                        | UN ~ [6, 10]                       |
| B      | Operating cost       | UN ~ [1, 3]                        | UN ~ [6, 10]                       |
| C      | $t_c$ times          | UN ~ [0.15, 0.25]                  | UN ~ [0.75, 1.25]                  |
|        | $t_l$ times          | UN ~ [0.4, 0.6]                    | UN ~ [2, 3]                        |
|        | $t_r$ times          | UN ~ [0.75, 0.95]                  | UN ~ [3.75, 4.25]                  |
| D      | Due date tightness   | Loose                              | Tight                              |
|        |                      | UN ~ [0.2, 0.8] · average makespan | UN ~ [0.1, 0.4] · average makespan |

Gnu C compiler. The problems were solved on a 400 Mhz UltraSPARC station.

In order to solve different problems at different difficulty levels, we determined four experimental factors that can affect the efficiency of the base heuristic. The factors can be seen in Table 1. The experimental design is a  $2^4$ -full factorial design with two levels for each factor. We take 5 replications for each factor combination, resulting in 80 randomly generated runs that will be solved by using PSGA with different parameters.

The number of operations, factor A, determines the size and load of the system. The operating cost, factor B, is used to calculate the machining and non-machining costs. Factor C, tool changing, loading and replacing times, determines the nonmachining times. The second and third factors are important for giving part loading and tool loading decisions, since they affect the machining and nonmachining cost terms in the first objective function and nonmachining times in the total weighted tardiness objective. Factor D, the due date tightness factor, is used to determine the due dates of the parts. When due dates are tight, the weighted tardiness of parts becomes significant after a certain time point. This increases the trade-off between two objectives. The average makespan in due date tightness factor is calculated as average makespan =  $((\sum_p \sum_m t_{pm}^m / M) + (N \cdot \text{average non-machining time})) / M$ . The average nonmachining times, which are affected by Factors A and C, are estimated as 1.25, 6, 2.5 and 12 for the factor combinations (0,0), (0,1), (1,0) and (1,1), respectively.

The other parameters of the problems are as follows. We have 3 nonidentical parallel CNC machines and 50 parts. The weights that show the relative importance of parts are selected from the integer interval UN ~ [1, 3]. The tool magazine

capacities of the machines are selected from UN ~ [10, 15]. There are ten different tool types and two tool alternatives for each operation. The parameters that are used to find optimal machining conditions are determined as follows. The horse-powers of machines are selected randomly from the interval UN ~ [3, 5]. The diameter of the generated surface,  $D_{pi}$ , and length of the surface,  $L_{pi}$ , for each operation are selected randomly from the interval UN ~ [1.5, 3] and UN ~ [4, 8], respectively. The surface finish requirement,  $SF_{pi}$ , and depth of cut,  $d_{pi}$ , for the finishing operation, which is the last operation for each part, are selected from distributions UN ~ [30, 70] and UN ~ [0.025, 0.075], respectively.  $SF_{pi} = \text{UN} \sim [300, 500]$  and  $d_{pi} = \text{UN} \sim [0.2, 0.3]$  for the roughing operations.

The first performance measure is the area covered by the nondominated solutions in the objective space. The minimum, average and maximum areas for all fitness assignment methods at different population size-generation levels are summarized in Table 2. When population size is 20 and maximum generation is 30, the average area covered by the nondominated solution set is 0.7331 for the first fitness assignment method, Linear, where weighted linear function of two objectives are taken. It is 0.7200 for Tchebycheff, 0.7277 for NSGA and 0.7401 for NSAPV. The proposed fitness assignment method, NSAPV, gives the largest average area for all levels. When population size is 20, Linear is the second best algorithm followed by the NSGA. As the population size and the number of maximum generations increase, NSGA gives better solutions than Linear since there is enough time for the convergence of NSGA. Tchebycheff method gives the worst results at all levels.

The second performance measure is the coverage difference of two sets, CD(A,B), which shows the

**Table 2.** Minimum, average and maximum values of area measure

| Levels     | Linear |        |        | Tchebycheff |        |        | NSGA   |        |        | NSAPV  |        |        |
|------------|--------|--------|--------|-------------|--------|--------|--------|--------|--------|--------|--------|--------|
|            | Min    | Avg    | Max    | Min         | Avg    | Max    | Min    | Avg    | Max    | Min    | Avg    | Max    |
| (20,30,5)  | 0.5705 | 0.7331 | 0.8791 | 0.5378      | 0.7200 | 0.8784 | 0.5632 | 0.7277 | 0.8794 | 0.5731 | 0.7401 | 0.8793 |
| (20,150,1) | 0.5666 | 0.7403 | 0.9962 | 0.5073      | 0.7340 | 0.8784 | 0.5535 | 0.7353 | 0.8808 | 0.5774 | 0.7504 | 0.8830 |
| (30,40,5)  | 0.5739 | 0.7475 | 0.8940 | 0.5701      | 0.7389 | 0.9021 | 0.5917 | 0.7557 | 0.8844 | 0.5956 | 0.7619 | 0.8963 |
| (30,200,1) | 0.5609 | 0.7565 | 0.9955 | 0.5655      | 0.7489 | 0.9391 | 0.5963 | 0.7659 | 0.9623 | 0.5903 | 0.7756 | 0.9988 |

**Table 3.** Coverage differences between the algorithms

| Levels             | Algorithms  | Linear | Tchebycheff | NSGA   | NSAPV  |
|--------------------|-------------|--------|-------------|--------|--------|
| (20,30,5)<br>(L1)  | Linear      |        | 0.0208      | 0.0162 | 0.0107 |
|                    | Tchebycheff | 0.0077 |             | 0.0078 | 0.0053 |
|                    | NSGA        | 0.0108 | 0.0155      |        | 0.0076 |
|                    | NSAPV       | 0.0177 | 0.0253      | 0.0200 |        |
| (20,150,1)<br>(L2) | Linear      |        | 0.0190      | 0.0202 | 0.0102 |
|                    | Tchebycheff | 0.0126 |             | 0.0140 | 0.0087 |
|                    | NSGA        | 0.0152 | 0.0153      |        | 0.0093 |
|                    | NSAPV       | 0.0202 | 0.0251      | 0.0244 |        |
| (30,40,5)<br>(L3)  | Linear      |        | 0.0180      | 0.0082 | 0.0057 |
|                    | Tchebycheff | 0.0094 |             | 0.0067 | 0.0049 |
|                    | NSGA        | 0.0164 | 0.0235      |        | 0.0084 |
|                    | NSAPV       | 0.0201 | 0.0279      | 0.0146 |        |
| (30,200,1)<br>(L4) | Linear      |        | 0.0176      | 0.0099 | 0.0054 |
|                    | Tchebycheff | 0.0100 |             | 0.0085 | 0.0041 |
|                    | NSGA        | 0.0194 | 0.0255      |        | 0.0088 |
|                    | NSAPV       | 0.0244 | 0.0308      | 0.0184 |        |

contribution of set A to the area covered by set B. The coverage difference between the fitness assignment methods at each level can be seen in Table 3. The contribution of Tchebycheff and NSGA to NSAPV is less than 1% at levels 1 and 2. At levels 3 and 4, the coverage difference of all algorithms over NSAPV is less than 1%. NSAPV is again the best algorithm for all levels. At levels 1 and 2, Linear can be taken as the second best algorithm.

The third performance measure is the expected

utility. Hansen and Jazskiewicz (1998) suggest that the Tchebycheff function should be used for aggregating the objectives when the decision maker's utility function is unknown. As we try to minimize weighted Tchebycheff function of two objectives, the smaller the expected utility the better the algorithm is. According to the results, which can be seen in Table 4, NSAPV gives the minimum utility at all levels. These results are consistent with the area measure.

The next performance measure, which is used to

**Table 4.** Minimum, average and maximum values of expected utility measure

| Levels     | Linear |        |        | Tchebycheff |        |        | NSGA   |        |        | NSAPV  |        |        |
|------------|--------|--------|--------|-------------|--------|--------|--------|--------|--------|--------|--------|--------|
|            | Min    | Avg    | Max    | Min         | Avg    | Max    | Min    | Avg    | Max    | Min    | Avg    | Max    |
| (20,30,5)  | 0.0660 | 0.1126 | 0.1517 | 0.0659      | 0.1175 | 0.2114 | 0.0660 | 0.1151 | 0.1809 | 0.0602 | 0.1117 | 0.1641 |
| (20,150,1) | 0.0125 | 0.1105 | 0.1540 | 0.0657      | 0.1124 | 0.1737 | 0.0597 | 0.1129 | 0.1822 | 0.0600 | 0.1082 | 0.1501 |
| (30,40,5)  | 0.0616 | 0.1086 | 0.1499 | 0.0515      | 0.1116 | 0.1889 | 0.0639 | 0.1072 | 0.1467 | 0.0545 | 0.1054 | 0.1465 |
| (30,200,1) | 0.0069 | 0.1059 | 0.1537 | 0.0276      | 0.1078 | 0.1527 | 0.0231 | 0.1043 | 0.1467 | 0.0039 | 0.1011 | 0.1471 |

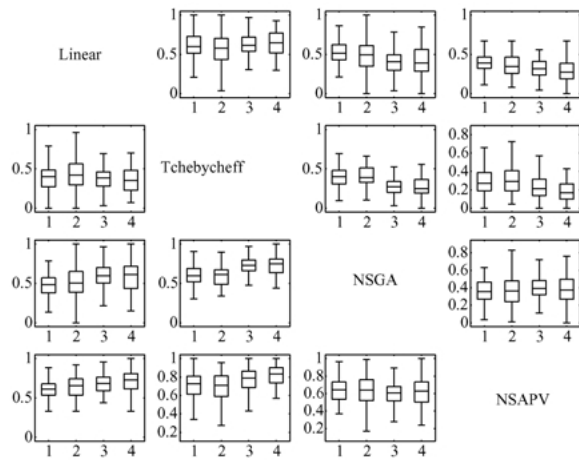


Fig. 4. Box-whisker plots of R1 measure for all algorithms at all levels.

make pairwise comparisons between two algorithms, is the probability that an algorithm gives a better solution than another algorithm,  $R1$ . The box-whisker plots representing the distribution of  $R1$  values for all methods at different population size-maximum generation levels can be seen in Fig. 4. In each rectangle, four box-whisker plots can be seen for levels (20,30,5), (20,150,1), (30,40,5) and (30,200,1), respectively. The box-whisker plots in each rectangle shows the probability that the algorithm A in the corresponding row gives a better solution than the algorithm B in the corresponding column,  $R1(A,B)$ . For example, the rectangle at the bottom left corner includes box-whisker plots for  $R1(NSAPV, Linear)$  at all levels. These box-whisker plots can also be used as a surrogate measure to identify the range of  $R1$  values. The top and bottom ends of the whiskers show the maximum and minimum values in the distribution, the top and bottom edges of the boxes show the 75th and 25th percentiles, and the lines going through the boxes show the median values or the 50th percentiles. According to the results, the probability that NSAPV

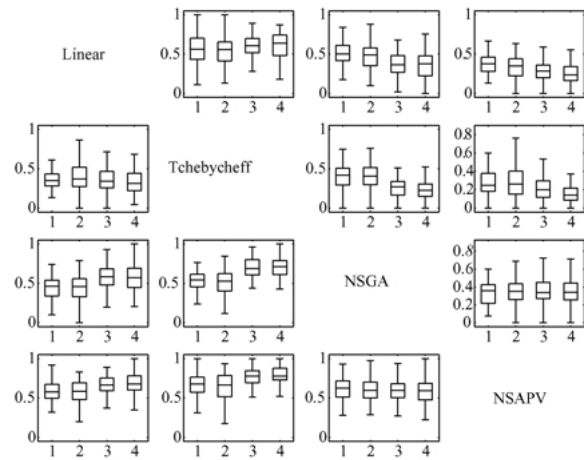


Fig. 5. Box-whisker plots of coverage measure for all algorithms at all levels.

gives better solutions than other algorithms is around 0.6–0.7. This shows that the proposed NSAPV method is clearly better than the others.

The last performance measure for comparing different fitness assignment and selection methods is the coverage between two sets. At all levels, NSAPV covers 60–65% of other algorithms, but other algorithms can cover only 25–35% of NSAPV as shown in Fig. 5. NSAPV is again the best algorithm according to the coverage measure. These results are consistent with the  $R1$  measure.

Up to now, we have analyzed the performance of the fitness assignment methods. We did not look at the difference between the single start and multistart runs. In order to see the difference, we use the coverage measure as an example. The coverage between the single start and multistart runs for all algorithms can be seen in Table 5. When the population size is 20, we can either have one long single run of 150 generations, denoted by L2, or 5 multiple short runs of 30 generations, denoted by L1. For the Linear function, 41.98% of the members of sets found by single start

Table 5. Coverage between single start and multiple start runs

| Algorithms  | Coverage (L1, L2) | Coverage (L2, L1) | Coverage (L3, L4) | Coverage (L4, L3) |
|-------------|-------------------|-------------------|-------------------|-------------------|
| Linear      | 0.4198            | 0.5803            | 0.4056            | 0.6128            |
| Tchebycheff | 0.4050            | 0.6085            | 0.4381            | 0.5917            |
| NSGA        | 0.3818            | 0.6263            | 0.3936            | 0.6267            |
| NSAPV       | 0.3591            | 0.5859            | 0.3228            | 0.6152            |

runs are covered by the members of multistart runs. The single start runs cover 58.03% of multistart runs. This means, if we have time to take longer runs, the algorithms could give better results, since the convergence is slow. We had to limit the maximum number of generations due to the computational complexity of the problem. In general, single start runs perform better than the multistart runs for the same population size. Finally, we look at the computation times for all algorithms. The average CPU times in seconds can be seen in Table 6. As the population size-generation level increases, the computation times increase as expected. There is not much difference between the run times of different algorithms.

In order to see the effect of elitism, we take runs at levels 1 and 2 with the proposed algorithm, NSAPV.

The cardinality of the elite set, which is directly copied to the next generation, is increased by inserting nondominated solutions from the external nondominated solution set, and it changes between two and four. The performance measures can be seen in Tables 7 and 8 for single start and multistart runs. For single start runs, there is not much difference among the three alternatives for the cardinality of the elite set. This shows that the members of the population is enough for convergence to the global Pareto-optimal set. There is no need to insert a nondominated solution from the external nondominated set. For multiple start runs, inserting a nondominated solution to the elite set might improve the performance of PSGA. As there is not enough time for convergence, bringing ‘‘good’’ solutions to the population makes the results better. However, as the number of solutions inserted to the

**Table 6.** CPU times in seconds

| <i>Levels</i> | <i>Linear</i> | <i>Tchebycheff</i> | <i>NSGA</i> | <i>NSAPV</i> |
|---------------|---------------|--------------------|-------------|--------------|
| (20,30,5)     | 1004          | 1003               | 1006        | 1001         |
| (20,150,1)    | 982           | 984                | 983         | 985          |
| (30,40,5)     | 1988          | 2000               | 1990        | 1983         |
| (30,200,1)    | 1954          | 1965               | 1957        | 1947         |

**Table 7.** Area and expected utility—elitism

| <i>Levels</i> | <i> eliteset </i> | <i>Area</i> |            |            | <i>Expected utility</i> |            |            |
|---------------|-------------------|-------------|------------|------------|-------------------------|------------|------------|
|               |                   | <i>Min</i>  | <i>Avg</i> | <i>Max</i> | <i>Min</i>              | <i>Avg</i> | <i>Max</i> |
| (20,150,1)    | 2                 | 0.5774      | 0.7507     | 0.8830     | 0.0600                  | 0.1082     | 0.1501     |
|               | 3                 | 0.5781      | 0.7489     | 0.8814     | 0.0589                  | 0.1092     | 0.1500     |
|               | 4                 | 0.5885      | 0.7490     | 0.8925     | 0.0596                  | 0.1093     | 0.1476     |
| (20,30,5)     | 2                 | 0.5731      | 0.7401     | 0.8793     | 0.0602                  | 0.1117     | 0.1641     |
|               | 3                 | 0.5731      | 0.7539     | 0.9733     | 0.0303                  | 0.1076     | 0.1544     |
|               | 4                 | 0.5646      | 0.7468     | 0.9113     | 0.0640                  | 0.1098     | 0.1547     |

**Table 8.** Coverage, coverage difference, R1—elitism

| <i>Levels</i> | <i> eliteset </i> | <i>Coverage</i> |        |        | <i>Coverage difference</i> |        |        | <i>R1</i> |        |        |
|---------------|-------------------|-----------------|--------|--------|----------------------------|--------|--------|-----------|--------|--------|
|               |                   | 2               | 3      | 4      | 2                          | 3      | 4      | 2         | 3      | 4      |
| (20,150,1)    | 2                 |                 | 0.4152 | 0.4153 |                            | 0.0170 | 0.0164 |           | 0.4667 | 0.4685 |
|               | 3                 | 0.5095          |        | 0.4681 | 0.0155                     |        | 0.0131 | 0.5333    |        | 0.4967 |
|               | 4                 | 0.5059          | 0.4616 |        | 0.0151                     | 0.0132 |        | 0.5315    | 0.5033 |        |
| (20,30,5)     | 2                 |                 | 0.3283 | 0.3514 |                            | 0.0090 | 0.0111 |           | 0.3984 | 0.4208 |
|               | 3                 | 0.5982          |        | 0.4610 | 0.0228                     |        | 0.0181 | 0.6016    |        | 0.5322 |
|               | 4                 | 0.5857          | 0.4350 |        | 0.0179                     | 0.0110 |        | 0.5792    | 0.4678 |        |

population increases, the quality of the nondominated solution set declines. Because, there will be less number of alternatives for the PSGA to generate different solutions in each generation.

As a summary, we can say that the proposed fitness assignment method gives better results than all other algorithms according to all performance measures. Linear is the second best algorithm when the population size-generation level is low. As the population size-generation level increases, NSGA becomes better than Linear. Combining all these methods and finding a ‘‘global’’ nondominated solution set is not a good alternative, since it increases the computation time. In addition, other algorithms cannot add much to the sets found by the NSAPV. The single start runs are better than multistart runs for convergence to the global Pareto-optimal set. The elitism can increase the performance of PSGA for multiple start runs. For single start runs, there is no need to take a solution from the external nondominated set and copy to the next generation. Because the fitness assignment and selection mechanism in PSGA is good enough for convergence to the global Pareto set, and copying solutions from an external set might adversely affect the evolutionary search process of the PSGA which is important to maintain diversity of the nondominated solution set.

## 6. Conclusions

In this study, we solved bicriteria tool management and scheduling problems simultaneously in FMS. This study is among few studies that consider these problems simultaneously. Most of the existing studies solve these two problems independently without considering the interaction between them. We have two objectives, minimizing the manufacturing cost and minimizing total weighted tardiness, which are in conflict with each other. We can decrease the machining cost by increasing the cutting speed or feed rate, but this increases the tooling and non-machining costs due to high utilization of tools and more frequent tool changes. The weighted tardiness increases or decreases according to the change in machining and nonmachining times. A PSGA is used to generate approximately efficient solutions for the bicriteria problem. This is the first study in which problem space search is used in solving a MOP for finding the nondominated solutions. This study is

important to highlight some of the difficulties that could be faced when PSGA is used in a MOP. The problems are about the convergence to the global Pareto-optimal set and the diversity of the nondominated solutions. Therefore, we proposed a new fitness assignment and selection method, which could be a trivial problem in single objective optimization problems, in order to find a well-diversified nondominated solution set that is close to the global Pareto-optimal set. The proposed method, NSAPV, is a composite measure and gives higher fitness values to the nondominated solutions which are closer to the global Pareto-optimal set, have better aggregated objective function value and less number of neighbors in objective space. The quality of the nondominated solution sets were evaluated by using the different performance measures and the proposed method gave the best solutions according to all performance measures.

## Acknowledgment

This work was supported in part by the NATO Collaborative Research Grant CRG-971489.

## Appendix: Notation

### Parameters:

|                                 |  |
|---------------------------------|--|
| $\alpha_j, \beta_j, \gamma_j$ : | speed, feed, depth of cut exponents for tool $j$   |
| $C_j$ :                         | Taylor’s tool life constant for tool $j$   |
| $d_{pi}$ :                      | depth of cut for operation $i$ of part $p$ (in)  |
| $D_{pi}$ :                      | diameter of the generated surface for operation $i$ of part $p$ (in)   |
| $L_{pi}$ :                      | length of the generated surface for operation $i$ of part $p$ (in)   |
| $DD_p$ :                        | due date of part $p$   |
| $w_p$ :                         | weight of part $p$   |
| $C_{om}$ :                      | operating cost of machine $m$ (\$/min)   |
| $C_{tj}$ :                      | cost of tool $j$ (\$/tool)   |
| $t_{cjm}$ :                     | tool interchange time of tool $j$ with the required tool for the next operation in machine $m$   |
| $t_{jm}$ :                      | time required to take a single tool $j$ from central tool storage and load on machine $m$ when there is a free slot on the tool magazine |

$t_{r_{jm}}$ : tool replacing time of worn tool  $j$  with a new tool from central tool storage to machine  $m$

### Decision variables

$v_{pijm}$ : cutting speed for operation  $i$  of part  $p$  using tool  $j$  on machine  $m$  (fpm)  
 $f_{pijm}$ : feed rate for operation  $i$  of part  $p$  using tool  $j$  on machine  $m$  (ipr)  
 $U_{pijm}$ : usage rate of tool  $j$  in the operation  $i$  of part  $p$  on machine  $m$   
 $T_{pijm}$ : tool life of tool  $j$  for operation  $i$  of part  $p$  on machine  $m$   
 $C_{pijm}^m$ : machining cost of operation  $i$  of part  $p$  using tool  $j$  on machine  $m$   
 $C_{pijm}^{nm}$ : nonmachining cost of operation  $i$  of part  $p$  using tool  $j$  on machine  $m$   
 $C_{pijm}^{tool}$ : tooling cost of operation  $i$  of part  $p$  using tool  $j$  on machine  $m$   
 $tard_{pm}$ : tardiness of part  $p$  on machine  $m$   
 $t_{m_{pijm}}$ : machining time of operation  $i$  of part  $p$  using tool  $j$  on machine  $m$   
 $t_{pm}^m$ : total machining time of part  $p$  on machine  $m$   
 $t_{pm}^{nm}$ : total nonmachining time of part  $p$  on machine  $m$   
 $t_m^{now}$ : current time on machine  $m$

### References

- Akturk, M. S. and Avci, S. (1996) Tool allocation and machining conditions optimization for CNC machines. *European Journal of Operational Research*, **94**(2), 335–348.
- Akturk, M. S. and Ozkan, S. (2001) Integrated scheduling and tool management in flexible manufacturing systems. *International Journal of Production Research*, **39**(12), 2697–2722.
- Bernardo, J. J. and Lin, K.-S. (1994) An interactive procedure for bi-criteria production scheduling. *Computers & Operations Research*, **21**(6), 677–688.
- Deb, K. and Goel, T. (2000) Controlled elitist non-dominated sorting genetic algorithms for better convergence. Technical Report 200004, Kanpur Genetic Algorithms Laboratory (KanGAL). (<http://www.iitk.ac.in/kangal/pub.htm>).
- Gray, A. E., Seidman, A. and Stecke, K. E. (1993) A synthesis of decision models for tool management in automated manufacturing. *Management Science*, **39**(5), 549–567.
- Hajela, P. and Lin, C.-Y. (1992) Genetic search strategies in multicriterion optimal design. *Structural Optimization*, **4**, 99–107.
- Hansen, M. P. and Jazskiewicz, A. (1998) Evaluating the quality of approximations to the non-dominated set. IMM Technical report, Technical University of Denmark. (<http://www-idss.cs.put.poznan.pl/~jazskiewicz/pub.html>).
- Jin, Y., Olhofer, M. and Sendhoff, B. (2001) Dynamic weighted aggregation for evolutionary multi-objective optimization: Why does it work and how? Technical report, Honda R&D Europe. ([http://www.optimization-online.org/DB\\_HTML/2001/03/297.html](http://www.optimization-online.org/DB_HTML/2001/03/297.html)).
- Schaffer, J. D. (1985) Multiple objective optimization with vector evaluated genetic algorithms, in *Proceedings of an International Conference on Genetic Algorithms and Their Applications*, Grefenstette J. J. (ed.), pp. 93–100.
- Srinivas, N. and Deb, K. (1994) Multiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation*, **2**(3), 221–248.
- Storer, R. H., Wu, S. D. and Vaccari, R. (1992) New search spaces for sequencing problems with application to job shop scheduling. *Management Science*, **38**(10), 1495–1509.
- Tiwari, M. K. and Vidyarthi, N. K. (2000) Solving machine loading problems in a flexible manufacturing system using a genetic algorithm based heuristic approach. *International Journal of Production Research*, **38**(14), 3357–3384.
- Turkcan, A., Akturk, M. S. and Storer, R. H. (2001) A problem space genetic algorithm to solve bicriteria scheduling in flexible manufacturing systems. Technical Report IEOR-0101, Bilkent University.
- Van Veldhuizen, D. A. and Lamont, G. B. (2000) Multiobjective evolutionary algorithms: Analyzing the state-of-the-art. *Evolutionary Computation*, **8**(2), 125–147.
- Zitzler, E. (1999) *Evolutionary algorithms for multiobjective optimization*. PhD thesis, Swiss Federal Institute of Technology Zurich. (<http://www.lania.mx/~ccoello/EMOO/EMOObib.html>).
- Zitzler, E. and Thiele, L. (1998) Multiobjective optimization using evolutionary algorithms—a comparative case study, in *Fifth International Conference on Parallel Problem Solving from Nature (PPSN-V)*, pages 292–301, Springer, Berlin. (<http://www.lania.mx/~ccoello/EMOO/EMOObib.html>).