Distributed Computing of Pareto-Optimal Solutions Using Multi-Objective Evolutionary Algorithms

Kalyanmoy Deb, Pawan Zope, and Abhishek Jain

Kanpur Genetic Algorithms Laboratory (KanGAL)
Indian Institute of Technology Kanpur
Kanpur, PIN 208 016, India
deb@iitk.ac.in
http://www.iitk.ac.in/kangal/pub.htm

KanGAL Report Number 2002008

Abstract. In this paper, we suggest an approach for finding multiple Pareto-optimal solutions with a distributed computing system. When the number of objective functions are more, the resulting Pareto-optimal set is large, thereby requiring a single processor multi-objective EA (MOEA) approach to use a large population size to be run for a large number of generations. However, the task of finding the complete Pareto-optimal front can be distributed among a number of processors, each pre-destined to find a particular region of the Pareto-optimal set. Based on the guided domination approach, here we propose a modified domination criterion for this task. The proof-of-principle results obtained with a parallel version of NSGA-II shows the efficacy of the proposed approach.

1 Introduction

The importance and efficacy of using multi-objective evolutionary algorithms (MOEAs) have been well established in the recent past [3, 2]. Not only there exist quite a few efficient MOEAs, there also exist a number of interesting applications [8]. However, like in the single-objective optimization studies, the computational time needed for solving multi-objective optimization is usually large in solving real-world optimization problems. In solving such computationally expensive problems, distributed computing with multiple processors are often used in the context of single-objective optimization. However, not much studies have been made in using distributed computing for multi-objective optimization. A good review of the existing studies can be found in [2].

In this paper, we suggest a parallel MOEA approach based on non-dominated sorting GA (or NSGA-II), which attempts to distribute the task of finding the entire Pareto-optimal front among participating processors. This way, each processor is destined to find a particular part of a convex Pareto-optimal front. With the help of a number of test problems, the efficacy of the proposed procedure is demonstrated.

2 Need for Distributed Computing in EMO

The task of optimization refers to comparison of a number of solutions before arriving at the optimal solutions. Since the evaluation of a solution can be time-consuming for real-world problems, the use of distributed computing has always been a major thrust in the area of optimization. Thus, it is needless to reiterate the importance of distributed computing in the case of multi-objective optimization. In fact, the importance is even greater in multi-objective optimization for the following reason.

With the increase of the number of objectives, the dimension of the true Pareto-optimal front increases. For a fixed number N of randomly created solutions in an objective space $f_i \in [0, 1]$, the proportion of solutions in the non-dominated front is plotted in Figure 1. The figure shows how quickly the size of the non-dominated front increases with the number of objectives. When the task is to find a widely-distributed set of solutions on the entire Pareto-optimal front, the

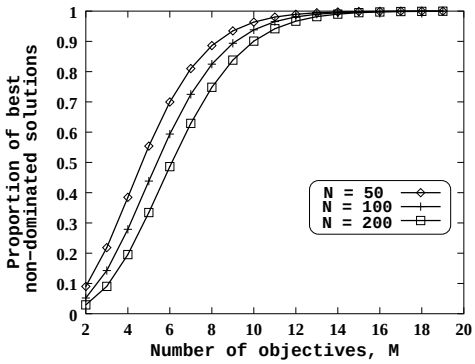

Fig. 1. The proportion of the best non-dominated solutions versus the number of objective functions

number of solutions needed to represent a large-dimensional surface is also large. This requires an MOEA to use a large population size, thereby increasing the overall computational time. Thus, it is all the more necessary to use a distributed computing environment for solving multi-objective problems.

A good description of different parallel MOEA implementations is presented in [2]. Most past studies have made a straightforward extension of parallel EA studies in solving multi-objective optimization problems. The studies mostly concentrate on three different approaches: (i) master-slave model, (ii) island model, and (iii) diffusion model. In the master-slave model, one master processor runs the MOEA and slave processors are used for evaluation purposes only. With such a model, the computational time can be expected to be reduced at most by P times with P processors. The solutions obtained by this procedure would be the same as that with a single processor, except that the computational time will be smaller. In the island model, different MOEAs are run on different processors and some solutions (called the migration rate) are migrated between processors after every few generations (called the migration frequency). The diffusion model is a fine-grained approximation of the island model. It is the island model on which our proposed approach is based. The island model provides a number of flexibilities to be tried:

1. First, each processor can be specifically assigned to search a particular portion of the entire search space. Care should be taken to ensure that no feasible search space is left out in the assignment for all processors. Since each processor works on a smaller search region, a computationally faster search is possible. However, it is not trivial to make such an assignment in an arbitrary problem and this is a serious bottleneck of this approach.
2. The above suggestion makes the search in each processor independent to each other. However, with the help of a migration plan each processor can be assigned to work on the entire search space and made to communicate their best solutions with other processors once in a while. This way, if some processors get stuck at some sub-optimal locations, the information of better solutions from other processors can break the stasis and help improve the proceedings. In the context of multi-objective optimization, since our goal is to find a number of solutions, there is no guarantee that each processor will find solutions in different parts of the Pareto-optimal region. Because of the overlapping search effort among multiple processors, the overall computational time may be large with such a model.
3. The migration policy can be used with scheme in which each processor is allocated to find a particular region of the Pareto-optimal front, although each processor searches the entire search space. Unlike the difficulties involved in systematically dividing the entire search space among multiple processors, the allocation of different portions of the Pareto-optimal front to different processors is not a difficult proposition. In such a scheme, care should be taken to

ensure that no feasible Pareto-optimal solution is left unallotted to any processor. A number of such schemes are certainly possible, and in the next section we discuss one such procedure for problems having convex Pareto-optimal fronts. Since each processor searches the entire search space, a migration plan will help each other, but since each processor is destined to find a particular portion of the search space, the agglomeration of solutions obtained by different processors will constitute a good diverse set of solutions.

Most past parallel MOEA studies based on the island model uses the second approach discussed above. Although some special action can be taken in each processor to ensure that there is a minimal overlap in the region searched by different processors, we do not pursue this approach further here. Instead, we suggest a procedure implementing the third concept discussed above.

3 Distributed Computing of Pareto-Optimal Solutions

Although not directly suggested for distributed computing of Pareto-optimal solutions, a number of techniques have been proposed for finding a part of the Pareto-optimal front, instead of the entire Pareto-optimal front with a single processor. In this section, we first briefly discuss a few such techniques and then suggest how one of them can be used for the distributed computing purpose, although other methods can also be used for the purpose.

3.1 Biased Sharing Approach

In order to bias a part of the Pareto-optimal front, the sharing approach used in many MOEAs (such as NSGA, MOGA, and others) can be modified:

$$d(i,j) = \left[\sum_{k=1}^{M} \frac{(f_k^{(i)} - f_k^{(j)})^2}{(f_k^{\max} - f_k^{\min})^2}\right]^{\frac{1}{2}}. \tag{1}$$

This distance metric is nothing but the normalized Euclidean distance between two objective vectors. In the proposed biased sharing approach [4], an unequal weightage is given to each objective in computing the Euclidean distance. For example, if w_k ($\in (0,1)$) is the weight assigned to the k-th objective function, then the normalized w'_k is calculated for a convex problem as follows:

$$w'_k = \frac{(1-w_k)}{\max_{m=1}^{M}(1-w_m)}, \tag{2}$$

and the modified distance metric is computed as follows:

$$d(i,j) = \left[\sum_{k=1}^{M} w'_k \frac{(f_k^{(i)} - f_k^{(j)})^2}{(f_k^{\max} - f_k^{\min})^2}\right]^{\frac{1}{2}}. \tag{3}$$

The fitness-based sharing can then be used with this distance metric. It is important to realize that when a sharing is performed with the second objective alone (with a large w_2), more solutions near the optimum value of f_1 would be obtained.

3.2 Weighted Domination Approach

Somewhat different from the above approach, Parmee et al. [7] suggested a weighted dominance principle. These investigators defined an index function $I_i(\mathbf{x}^{(1)}, \mathbf{x}^{(2)})$ between two solutions $\mathbf{x}^{(1)}$ and $\mathbf{x}^{(2)}$ for the i-th objective function as follows:

$$I_i(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}) = \begin{cases} 1, f_i(\mathbf{x}^{(1)}) \leq f_i(\mathbf{x}^{(2)}); \\ 0, \text{otherwise.} \end{cases} \tag{4}$$

Now, if solution $\mathbf{x}^{(1)}$ dominates solution $\mathbf{x}^{(2)}$, then each I_i function would take a value equal to one. In other words, we can write the usual condition for domination in a different way:

$$\sum_{i=1}^{M} I_i(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}) = M, \tag{5}$$

with the condition that $f_i(\mathbf{x}^{(1)}) < f_i(\mathbf{x}^{(2)})$ for at least one objective i. In the original definition of domination, all objectives are given equal importance. Rewriting the above equation and extending the relationship to inequality, one may write:

$$\sum_{i=1}^{M} \frac{1}{M} I_i(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}) \geq 1. \tag{6}$$

Generalizing the above concept for a weight vector $\mathbf{w}$ (such that $\sum_{i=1}^{M} w_i = 1$) which indicates a preference relationship among different objectives, we can write the above inequality for the $\mathbf{w}$-dominating condition as follows:

$$\sum_{i=1}^{M} w_i I_i(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}) \geq 1. \tag{7}$$

If all objectives are of equal importance, each $w_i = 1/M$ and we have the inequality shown in equation (6). However, for any other generic weight vector, we have the above condition for a solution $\mathbf{x}^{(1)}$ to be $\mathbf{w}$-dominating solution $\mathbf{x}^{(2)}$. Generalizing further, these investigators suggested the condition for a $(\mathbf{w}, \tau)$-dominance (with $\tau \leq 1$) between two solutions, as follows:

$$\sum_{i=1}^{M} w_i I_i(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}) \geq \tau. \tag{8}$$

Based on these conditions for $\mathbf{w}$-dominance and $(\mathbf{w}, \tau)$-dominance, a corresponding non-dominated and a Pareto-optimal set can also be identified. However, if the above weak dominance conditions are used, the obtained set would be a strict non-dominated one. However, if any one of the index functions is restricted for the strict inequality condition, a weak non-dominated front will be found.

3.3 Guided Domination Approach

In this approach [1], a weighted function of the objectives is defined as follows:

$$\Omega_i(\mathbf{f}(\mathbf{x})) = f_i(\mathbf{x}) + \sum_{j=1, j\neq i}^{M} a_{ij} f_j(\mathbf{x}), \quad i = 1, 2, \ldots, M. \tag{9}$$

where a_{ij} is the amount of gain in the j-th objective function for a loss of one unit in the i-th objective function. The above set of equations require fixing the matrix $\mathbf{a}$, which has a one in its diagonal elements. Now, we define a different domination concept for minimization problems as follows.

Definition 1 *A solution $\mathbf{x}^{(1)}$ dominates another solution $\mathbf{x}^{(2)}$, if $\Omega_i(\mathbf{f}(\mathbf{x}^{(1)})) \leq \Omega_i(\mathbf{f}(\mathbf{x}^{(2)}))$ for all $i = 1, 2, \ldots, M$ and the strict inequality is satisfied at least for one objective.*

Let us illustrate the concept for two ($M = 2$) objective functions, as we shall modify this approach for distributed computing of Pareto-optimal solutions. The two weighted functions are as follows:

$$\Omega_1(f_1, f_2) = f_1 + a_{12} f_2, \tag{10}$$

$$\Omega_2(f_1, f_2) = a_{21} f_1 + f_2. \tag{11}$$

The above equations can also be written as

$$\mathbf{\Omega} = \begin{bmatrix} 1 & a_{12} \\ a_{21} & 1 \end{bmatrix} \mathbf{f}, \qquad \text{or, } \mathbf{\Omega} = \mathbf{af}. \tag{12}$$

Figure 2(b) shows the contour lines corresponding to the above two linear functions passing through a solution A in the objective space. All solutions in the hatched region are dominated

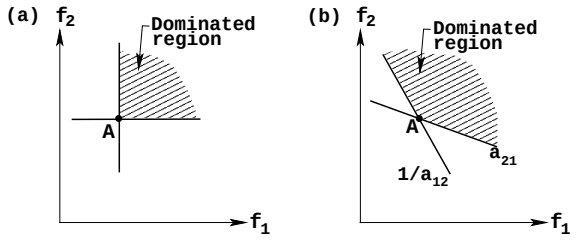

Fig. 2. The regions dominated by solution A: (a) the dominated region using the usual definition; (b) the dominated region using definition 1.

by A according to the above definition of domination. It is interesting to note that when using the usual definition of domination (Figure 2(a)), the region marked by a horizontal and a vertical line will be dominated by A. Thus, it is clear from these figures that the modified definition of domination allows a larger region to become dominated by any solution than the usual definition. It is also interesting to realize that since a larger region is now dominated, the complete Pareto-optimal front (as per the original domination definition) may not be non-dominated according to this new definition of domination. For the same value of the matrix **a** in the two-objective function illustration (Figure 2(b)), the resulting non-dominated front is depicted in Figure 3. This figure shows that regions near the individual champions are now not non-dominated. It is clear from this figure that some portion (shown by a thin continuous curve) of the Pareto-optimal region is dominated by a member in the middle portion of the Pareto-optimal region (shown by a bold curve). Thus, an MOEA is expected to find only the middle portion of the Pareto-optimal region, thereby biasing the search towards a particular region of the Pareto-optimal front. Thus, by choosing appropriate values for the elements of the matrix **a**, a part of the Pareto-optimal region can be emphasized. In the following subsection, we extend this guided domination approach for a distributed computing of Pareto-optimal solutions.

3.4 Proposed Approach

The idea of distributed computing is to allocate a processor a task of finding a particular region of the Pareto-optimal front. However, while distributing the task, the following three aspects should be keep in mind:

1. No Pareto-optimal solution is left to be discovered by all processors,

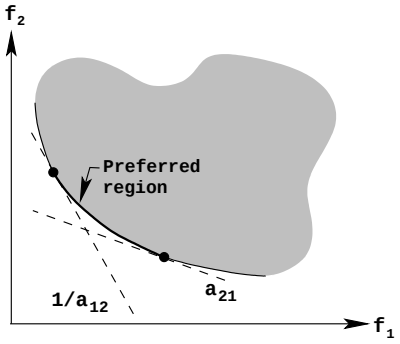

Fig. 3. The non-dominated portion of the Pareto-optimal region.

2. The overlap of solutions allocated by any two processors should be as small as possible for optimum allocation of computing resources, and
3. The procedure must be easy to extend for any number of objectives.

While using the guided domination approach for this purpose, the parameters a_{ij} must be chosen so as to satisfy the above aspects. The first two aspects can be ensured by choosing appropriate **a**-matrices for different processors. For example, for a two-objective problem, the following two **a**-matrices can be used

$$\begin{bmatrix} 1 & 0 \\ a & 1 \end{bmatrix}, \qquad \begin{bmatrix} 1 & 1/a \\ 0 & 1 \end{bmatrix}$$

for two processors. The slope a can be any positive real number. However, the third aspect mentioned above is difficult to achieve using the guided domination approach directly. For example, with three objectives, the **a**-matrix has 3×3 elements and six different off-diagonal entries in the matrix need to be fixed. Moreover, since in this case the modified Pareto-optimal front would be bounded by tangential planes, instead of lines, it becomes difficult to determine the slopes a_{ij} of the planes. In the following, we suggest a modified technique for this purpose.

Let us consider a feasible objective space bounded by a circle, as shown in Figure 4. Let us also consider two other axes Ω_1 and Ω_2 inclined to the objective axes. The direction vectors of these two inclined axes are given as follows:

$$\eta_1 = d_1^{(1)}\hat{e}_1 + d_2^{(1)}\hat{e}_2, \tag{13}$$

$$\eta_2 = d_1^{(2)}\hat{e}_1 + d_2^{(2)}\hat{e}_2, \tag{14}$$

where $d^{(i)}$ is the direction cosine vector of η_i and $\hat{e}_i$ is the i-th coordinate direction. Now, any point $(f_1^{(p)}, f_2^{(p)})$ on the circle can also be represented by the Ω-coordinate system as $(\Omega_1^{(p)}, \Omega_2^{(p)})$ and we have the identity:

$$\begin{aligned} f_1^{(p)}\hat{e}_1 + f_2^{(p)}\hat{e}_2 &= \Omega_1^{(p)}\eta_1^{(p)} + \Omega_1^{(p)}\eta_2^{(p)}, \\ &= \left(\Omega_1^{(p)}d_1^{(1)} + \Omega_2^{(p)}d_1^{(2)}\right)\hat{e}_1 + \left(\Omega_1^{(p)}d_2^{(1)} + \Omega_2^{(p)}d_2^{(2)}\right)\hat{e}_2. \end{aligned}$$

Comparing terms for $\hat{e}_i$, we can write the following matrix equation:

$$\begin{bmatrix} f_1^{(p)} \\ f_2^{(p)} \end{bmatrix} = \begin{bmatrix} d_1^{(1)} & d_1^{(2)} \\ d_2^{(1)} & d_2^{(2)} \end{bmatrix} \begin{bmatrix} \Omega_1^{(p)} \\ \Omega_2^{(p)} \end{bmatrix}, \tag{15}$$

$$\text{or,} \quad \mathbf{f} = \mathbf{T\Omega}. \tag{16}$$

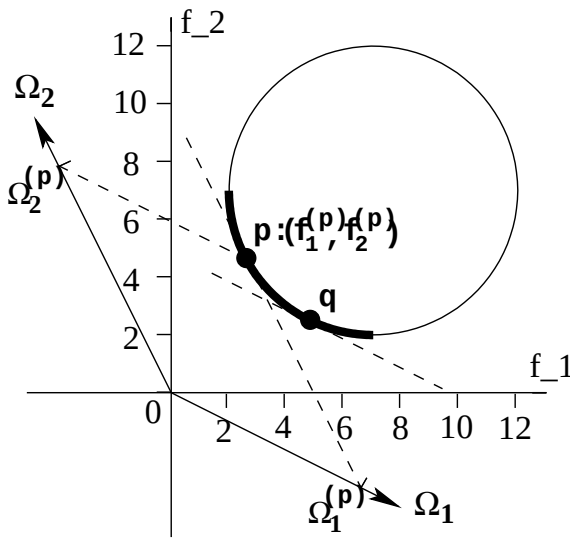

Fig. 4. The f_1-f_2 objective space.

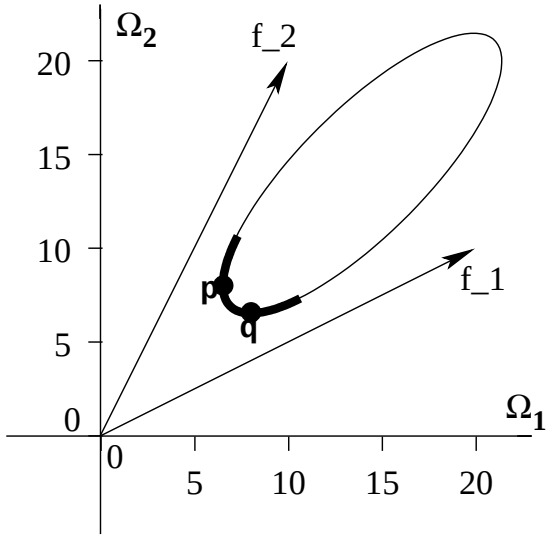

Fig. 5. The Ω_1-Ω_2 objective space.

From the above relationship, we can write

$$\mathbf{\Omega} = \mathbf{T}^{-1}\mathbf{f}. \tag{17}$$

Now relating the direction cosine vectors with respect to a_{ij}, we have

$$d^{(1)} = \frac{1}{\sqrt{1+a_{21}^2}}\left(1, -a_{21}\right)^T,$$
$$d^{(2)} = \frac{1}{\sqrt{1+a_{12}^2}}\left(-a_{12}, 1\right)^T.$$

Forming the $\mathbf{T}$ matrix and taking the inverse, we obtain

$$\begin{bmatrix} \frac{1-a_{12}a_{21}}{\sqrt{1+a_{21}^2}}\Omega_1^{(p)} \\ \frac{1-a_{12}a_{21}}{\sqrt{1+a_{12}^2}}\Omega_2^{(p)} \end{bmatrix} = \begin{bmatrix} 1 & a_{12} \\ a_{21} & 1 \end{bmatrix} \begin{bmatrix} f_1^{(p)} \\ f_2^{(p)} \end{bmatrix}. \tag{18}$$

The Ω_1-Ω_2 space for the f_1-f_2 objective space is shown in Figure 5. The region pq on the original circle is bounded by tangents with Ω_1 and Ω_2 direction vectors. Thus, for the transformed objectives, the entire Pareto-optimal front is bounded within the arc pq. The transformation of the objective space and the location of the new Pareto-optimal solutions are clearly shown in the figure. Comparing this equation with equation 12, we observe that the right sides are equal and the Ω-axes are multiplied by different numbers. However, it is interesting to note that multiplying Ω_1 and Ω_2 values by different non-zero constants may translate or magnify the Pareto-optimal front in the Ω_1-Ω_2 space, but it does not change the Pareto-optimal solutions in terms of f_1 and f_2 values. Thus, the original **a**-matrix can be replaced with the $\mathbf{T}^{-1}$ matrix, constructed simply from the direction cosines of the two inclined lines. Here is the procedure:

1. Form the **a**-matrix from direction cosines of transformed axes,
2. Calculate the inverse of the **a** matrix and use domination test with the transformed objectives (with $\mathbf{\Omega}$ vector calculated using equation 17).

In order to ensure the first property of not leaving any feasible solution by the mapping, we need to ensure that two processors share a common direction. In Figure 6, we show the allocation plan for choosing direction cosines for 1, 2, 3, and 4 processors in a two-objective problem. The intersecting point of the two adjacent axes directions indicate the allocation plan of a particular processor. Note that the left-most plot indicates a single processor with $\Omega_1 = f_1$ and $\Omega_2 = f_2$.

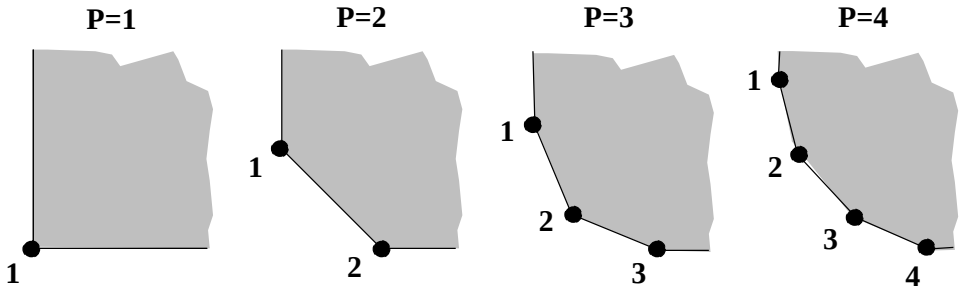

Fig. 6. Allocation plan for two-objective problems.

The next figure indicates that two processors share one common direction (the inclined direction), and so on. Although different schemes can be adopted for assigning common inclined directions, we suggest a methodology by which equal inscribed angles are assigned to each processor. For P processors solving a two-objective optimization problem, the included angle (in radians) between any two adjacent direction vectors is

$$\phi = \pi \left(1 - \frac{1}{2P}\right). \tag{19}$$

Extension to higher objectives Interestingly, the above argument can be extended to more than two dimensions and can be applied easily to any number of objectives. For example, in a three-objective space, the entire Pareto-optimal front can be determined by tangential properties of three planes: f_1-plane, f_2-plane and f_3-plane. If three arbitrary (but non-identical) directions (Ω_1, Ω_2, Ω_3) are chosen, as shown in Figure 7, the modified Pareto-optimal front will be bounded by the three new planes: Ω_1-Ω_2 plane, Ω_2-Ω_3 plane, and Ω_3-Ω_1 plane.

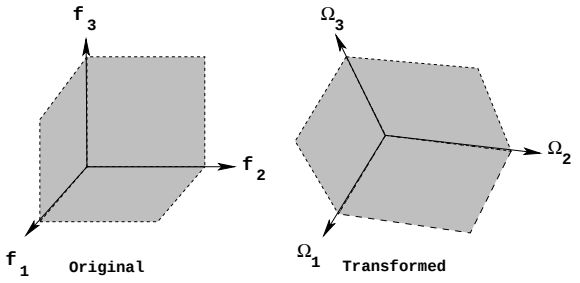

Fig. 7. Original and transformed planes.

The procedure of generating an allocation plan for two objectives illustrated in Figure 6 can be considered as successive 'chopping-off' of the bottom-left corner (or origin) of a square by straight

edges. Each corner at the bottom-left part of the square corresponds to the a processor for which the direction cosines are computed from the straight edges forming the corner. When an existing corner is chopped off, two new corners are added, thereby allowing one more processor to be used. For example, the $P = 2$ case is obtained from $P = 1$ case by chopping off the right-angle corner. Larger P cases can be considered as further chopping off the bottom-left corner of the square. The only restriction to the chopping process for P processors is that there must be exactly P corners generated by the chopping process.

This chopping principle can be extended for higher objectives, except that the bounded square now becomes a larger-dimensional hyper-box, and hyper-planes are used to chop-off the origin. Figures 8 to 11 show a number of allocation plans for different number of processors in a three-objective spherical Pareto-optimal front. Although the systematic chopping of each corner will

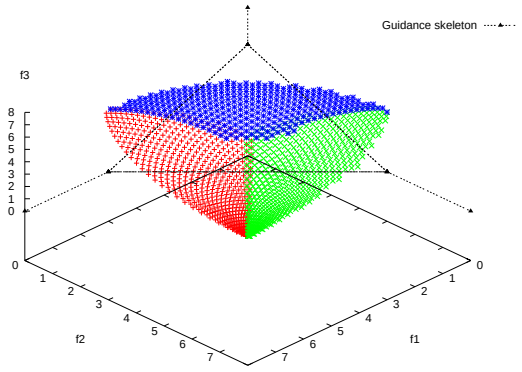

Fig. 8. 3 parts of Pareto-optimal front

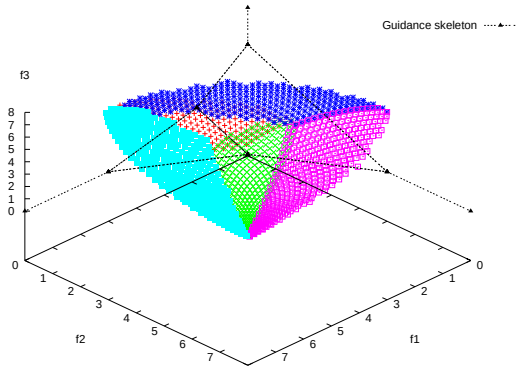

Fig. 9. 5 parts of Pareto-optimal front

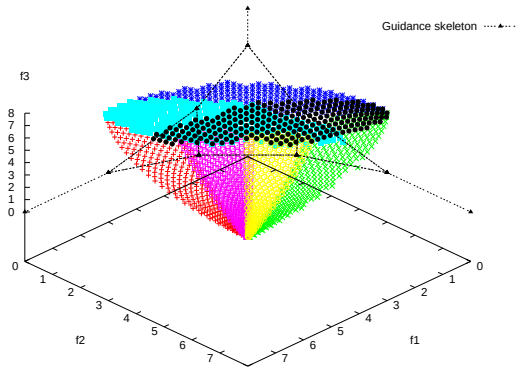

Fig. 10. 7 parts of Pareto-optimal front

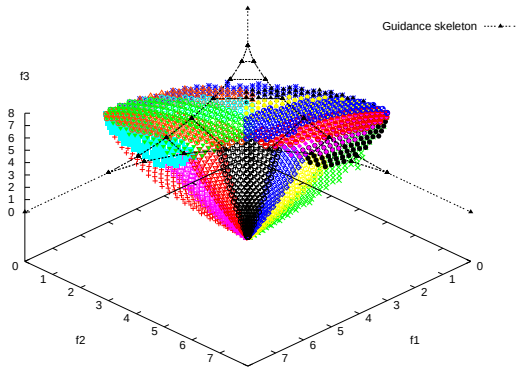

Fig. 11. 21 parts Pareto-optimal front

generate $P = (M - 1)k + 1$ corners (where k is any non-negative integer), Figures 12 and 13 also show some interesting allocation plans for different numbers of processors.

Need for migration In this study, we have started each processor with a population initialized from the same bounded region. In order to keep the total number of function evaluations the same, we have also kept a constant combined population size. Thus, for a study with P processors, we have used N/P population members in each processor, where N is the population size used for the single-processor run. With this in mind, it then becomes difficult for each processor to

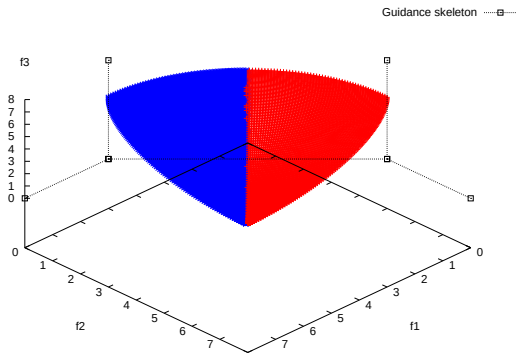

Fig. 12. 2 parts of Pareto-optimal front

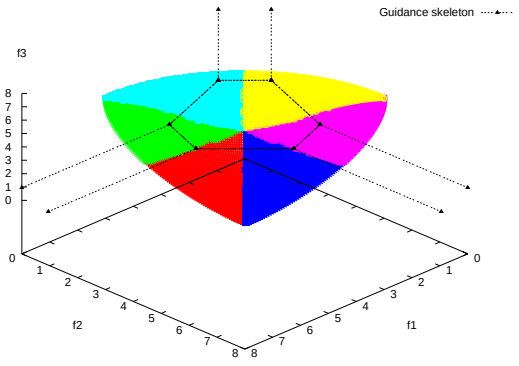

Fig. 13. 6 parts of Pareto-optimal front

overcome identical hurdles to reach to the Pareto-optimal front with a smaller population size. In order to get assistance from other processors in proceeding towards the Pareto-optimal front, we have introduced a migration policy among the processors. A certain number of solutions (migration rate) from one processor are sent to each other processor after every a few generations (migration frequency). Each processor is chosen systematically for its turn to migrate solutions to other processors. The new solutions replace a randomly chosen set of members from the existing population. Such information introduced by other processors may help a temporarily stagnant (or slow progressing) processor.

4 Simulation Studies

In this section, we present simulation results of the proposed parallel MOEA to a number two and three-objective test problems.

4.1 Two-Objective Test Problems

First, we study the 30-variable ZDT1 test problem [3]:

$$\begin{aligned} f_1(\mathbf{x}) &= x_1, \\ f_2(\mathbf{x}) &= g(\mathbf{x})h(f_1, g), \\ g(\mathbf{x}) &= 1 + \textstyle\sum_{i=2}^{30}(x_i - 0.5)^2, \\ h(f_1, g) &= 1 - \sqrt{f_1/g}. \end{aligned} \tag{20}$$

Here, each x_i lies $[0, 1]$. Figure 14 shows the hyper-area [9] for a serial (with one processor) NSGA-II run with a population size of 200. The simulated binary crossover operator (with $\eta_c = 10$ and probability 0.9) and polynomial mutation operator (with $\eta_m = 50$ and probability $1/30$) are used [3]. With a reference point taken as $(1.0646, 1.0646)$, the maximum hyper-area calculated for an infinite population lying on the true Pareto-optimal front is 0.8. The NSGA-II run is terminated when an hyper-area of 0.794 is achieved. In the particular run shown in Figure 14, 57 generations were needed to achieve this hyper-area.

Figure 15 shows the distribution of parallel real-parameter NSGA-II with two processors. The following two transformation matrices are used for the two processors:

$$\begin{bmatrix} 1 - \cos(0.75\pi) & \\ 0 & \sin(0.75\pi) \end{bmatrix}, \qquad \begin{bmatrix} \cos(0.75\pi) & 0 \\ -\sin(0.75\pi) & 1 \end{bmatrix}.$$

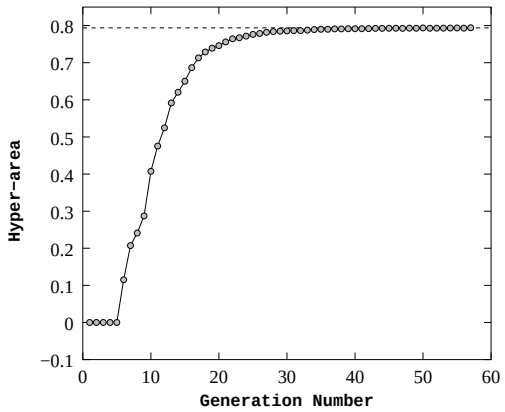

Fig. 14. Hyper-area calculated with (1.0646,1.0646) as the reference point.

Each processor is initialized with a population of size 100. Other GA parameters are kept the same as in the serial NSGA-II run. In this case, 30 solutions from a processor are migrated after every 5 generations to the other processor. At the end of 52 generations, the hyper-area calculated for the combined population of two processors is 0.794. Thus, it is clear that a similar distribution can be achieved by the parallel two-processor NSGA-II with more or less similar number of function evaluations.

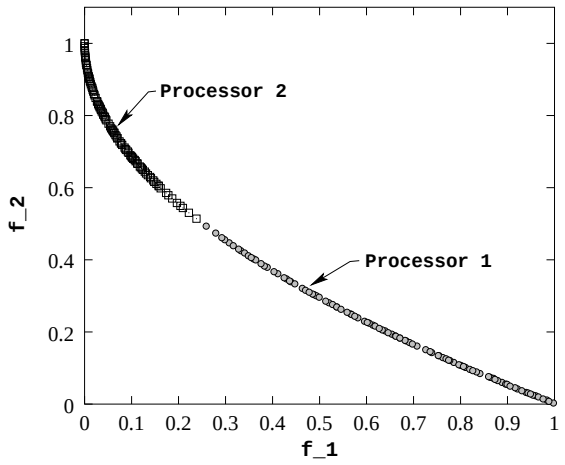

Fig. 15. Obtained solutions with two processors for ZDT1.

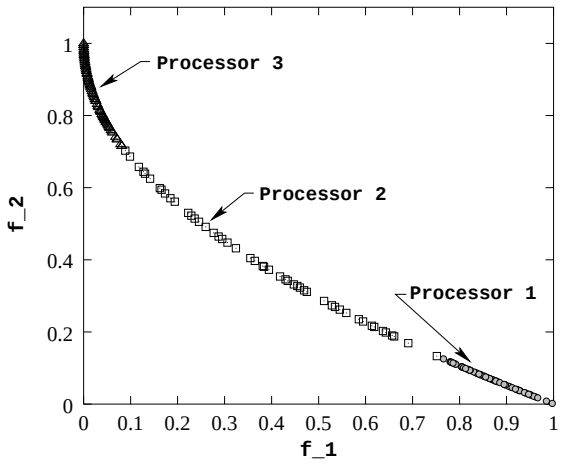

Fig. 16. Obtained solutions with three processors for ZDT1.

Figure 16 shows the distribution of solutions (with a hyper-area of 0.794) obtained with a three-processor NSGA-II after 57 generations. The following three transformation matrices are used:

$$\begin{bmatrix} 1 & -\cos(\frac{5}{6}\pi) \\ 0 & \sin(\frac{5}{6}\pi) \end{bmatrix}, \quad \begin{bmatrix} \cos(\frac{5}{6}\pi) & -\cos(\frac{2}{3}\pi) \\ -\sin(\frac{5}{6}\pi) & \sin(\frac{2}{3}\pi) \end{bmatrix}, \quad \begin{bmatrix} \cos(\frac{2}{3}\pi) & 0 \\ -\sin(\frac{2}{3}\pi) & 1 \end{bmatrix}.$$

In this case, 30 solutions from a processor is sent to two other processors after every 5 generations.

In order to investigate the effect of the migration policy on the distribution of solutions, we have performed a detailed parametric study on the ZDT1 test problem. The outcome of this study is

presented in Table 1. In each case, the average and standard deviation of the number of generations

Table 1. Parametric study of migration rate and frequency on ZDT1 to achieve a hyper-area of 0.794.

Migration Rate	Migration Frequency	Required Generations					
		Single Processor		Two Processors		Three Processors	
		Average	Std. Dev.	Average	Std. Dev.	Average	Std. Dev.
0	0	58.9	3.91	66.1	9.63	116.3	45.71
10	5			50.1	3.05	**59.6**	5.10
10	10			52.5	4.34	66.3	6.10
20	5			48.9	1.51	60.3	3.52
20	10			51.2	3.57	60.8	3.43
30	5			**48.3**	1.35	59.8	3.66
30	10			50.6	2.46	63.8	3.12
30	15			55.8	5.31	68.7	4.88
40	10			49.3	3.13	61.7	4.43
30	*5*			*57.2*	*2.99*	*70.2*	*4.58*

needed to achieve a hyper-area of 0.794 in ten runs are calculated. The first row shows the single processor result and multi-processor results without any migration. It is observed that the parallel NSGA-II does not perform well without migration. Although the outcome of the parallel NSGA-II depends on the chosen migration rate and frequency, most of the two-processor runs achieved the required hyper-area (meaning that a similar combined convergence and diversity measure is achieved) in a smaller number of generations than that required with a single processor NSGA-II. The best performance occurs when 30 solutions are migrated after every 5 generations. In this case, on an average two-processor NSGA-II requires about 18% lesser function evaluations to achieve a similar performance. This means that the use of two processors reduce the computation time by more than two times.

With three processors, however, the required number of generations to achieve a similar performance is slightly more. Although the best performance is achieved when 10 solutions are migrated after every 5 generations, the runs with migration rate of 30 and frequency 5 (as that obtained to be the best in the two-processor case) is also very close to this optimum performance. In the three-processor case, each processor is allocated 68 population members. Since all processors tackle the same ZDT1 problem (but each using a different dominance relationship), the population size allocated to each processor may not be enough for it to reach near the Pareto-optimal solutions. For every problem, there exists an optimal population size (even with a migration policy) below which the NSGA-II may not be able to reach the desired goal. For ZDT1, the two-processor NSGA-II with 100 population members in each processor seems to have produced an optimal performance.

The last row in the table also shows the performance of an island three-processor NSGA-II without the transformation procedure. Here, all three processors search the entire Pareto-optimal front, but 30 solutions are migrated at every 5 generations. Since all processors independently find solutions on the Pareto-optimal front, the agglomerated set of obtained solutions is not guaranteed to have adequate diversity. It is clear from the table that this procedure requires more generations to achieve an identical hyper-area of 0.794 than the proposed transformation approach with both two and three processors.

Figure 17 shows the obtained agglomerated solutions with four processors, each using 50 population members. The allocation plan is set according to equation 19 and is also shown in the figure. A migration rate of 20 and migration frequency of 5 are used. It is observed that only three processors (2 to 4) have found multiple solutions and only one Pareto-optimal solution ($f_1 = 1$, $f_2 = 0$) is found by the processor 1. We observe that the Pareto-optimal front follows $f_2 = 1 - \sqrt{f_1}$

for $f_1 \in [0, 1]$. At $f_1 = 1$, the slope of the Pareto-optimal front is -0.5, equivalent to an angle of 153.43 degrees from the f_1 axis. With the allocation plan shown in Figure 17, this extreme solution is captured by the second processor and there is no new solution left in the Pareto-optimal front for the first processor to find. When we change the direction cosines with an allocation plan as in Figure 18, solutions in all four processors are found. Here, 20 solutions are migrated after every 10 generations. The slope of the two axes lines for processors 1, 2, 3, and 4 are (26.565, 42.515), (42.515, 58.375), (58.375, 74.235), and (74.235, 90) degrees from the negative f_1 axis, respectively.

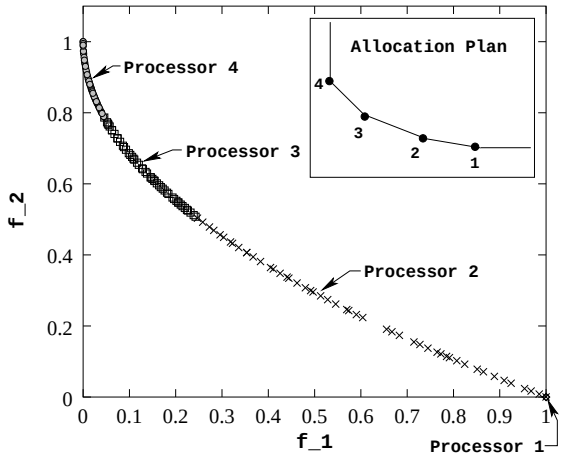

Fig. 17. Obtained solutions with four processors for ZDT1.

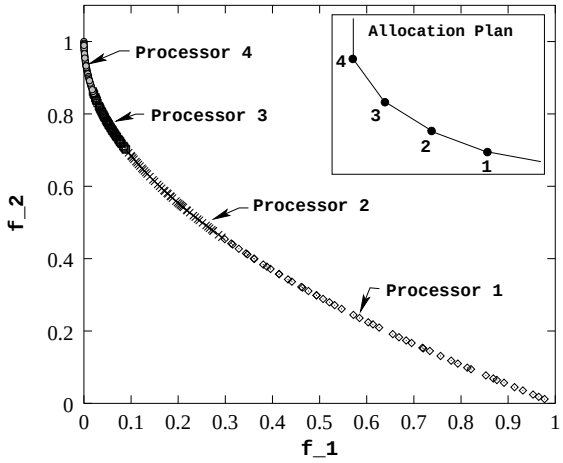

Fig. 18. Obtained solutions with four processors for ZDT1 with a different allocation plan.

Test Problem ZDT4 Next, we solve the test problem ZDT4 with $n = 10$ variables:

$$\begin{aligned} f_1(\mathbf{x}) &= x_1, \\ f_2(\mathbf{x}) &= g(\mathbf{x})h(f_1, g), \\ g(\mathbf{x}) &= 1 + 10(n-1) + \textstyle\sum_{i=2}^{n}(x_i^2 - 10\cos(4\pi x_i)), \\ h(f_1, g) &= 1 - \sqrt{f_1/g}. \end{aligned} \tag{21}$$

Here, x_1 lies $[0, 1]$ and all other variables x_i lie in $[-5, 5]$. Figure 19 shows the obtained solutions with two processors with each having 100 population members. The allocation plan is based in equation 19. Here, 30 population members are migrated in every 5 generations. The figure shows that the two processors combined has found the entire Pareto-optimal front.

Figure 20 shows the results with three processors. This time, each processor is run with 70 solutions each and 40 population members are migrated in every 10 generations. Once again, the three processors combined have found the entire Pareto-optimal front. Since the Pareto-optimal front for ZDT4 is identical to that in ZDT1 and since identical allocation plans are chosen in both experiments, very similar plots are obtained (compare Figure 15 with 19 and Figure 16 with 20).

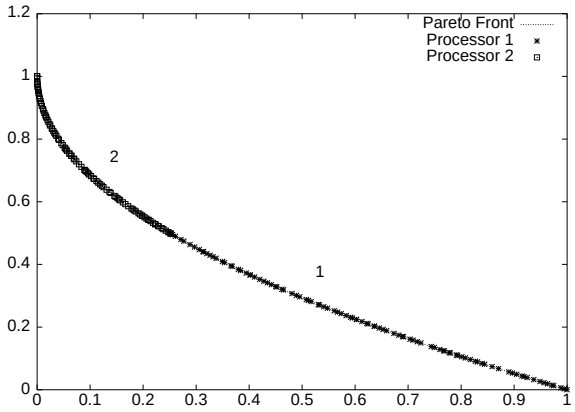

Fig. 19. Obtained non-dominated solution with 2-processor Parallel NSGA-II on ZDT4 problem.

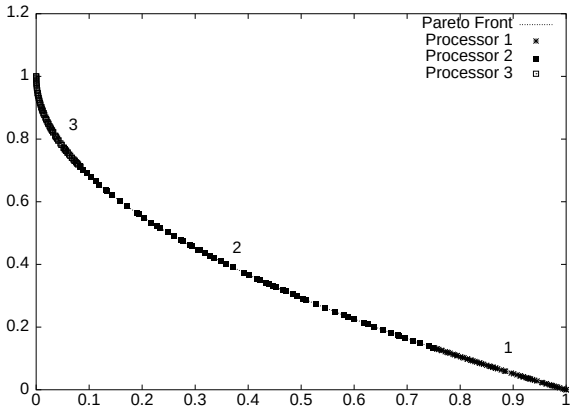

Fig. 20. Obtained non-dominated solution with 3-processor Parallel NSGA-II on ZDT4 problem.

Test Problem CTP7 The CTP7 test problem is a constrained test problem which makes the Pareto-optimal front a combination of a number of disconnected regions [3]:

$$\begin{aligned} &\text{Minimize } f_1(\mathbf{x}) = x_1, \\ &\text{Minimize } f_2(\mathbf{x}) = g(\mathbf{x}) \left(1 - \frac{f_1(\mathbf{x})}{g(\mathbf{x})}\right), \\ &\text{subject to } C(\mathbf{x}) \equiv \cos(\theta)[f_2(\mathbf{x}) - e] - \sin(\theta) f_1(\mathbf{x}) \geq \\ &\qquad a \left|\sin\left\{b\pi \left[\sin(\theta)(f_2(\mathbf{x}) - e) + \cos(\theta) f_1(\mathbf{x})\right]^c\right\}\right|^d. \end{aligned} \tag{22}$$

The following parameter values are used:

$$\theta = -0.05\pi, \quad a = 40, \quad b = 5, \quad c = 1, \quad d = 6, \quad e = 0.$$

Each of the five variables x_i lies in $[0, 1]$. Figure 21 shows the obtained front with a single processor. It is clear that not all regions are found by the NSGA-II with a population size of 200. This result agrees with the original result reported in [5]. However, when NSGA-II is run with three processors, each processor has a shorter Pareto-optimal region to discover. As a result, the parallel NSGA-II (with processor-wise population of 70) is able to find all the disconnected Pareto-optimal regions (Figure 22). Here, 30 solutions are migrated after every 5 generations. This example illustrates how the proposed approach can also be used in problems having a disconnected set of Pareto-optimal regions.

4.2 Three-Objective Test Problems

Next, we consider two three-objective test problems.

DTLZ2 test problem The first one is a modified DTLZ2 test problem having $n = 12$ variables:

$$\begin{aligned} &\text{Minimize } f_1(\mathbf{x}) = 2 - (1 - g(\mathbf{x})) \cos(x_1\pi/2) \cos(x_2\pi/2), \\ &\text{Minimize } f_2(\mathbf{x}) = 2 - (1 - g(\mathbf{x})) \cos(x_1\pi/2) \sin(x_2\pi/2), \\ &\text{Minimize } f_2(\mathbf{x}) = 2 - (1 - g(\mathbf{x})) \sin(x_1\pi/2), \\ &\quad \text{where } g(\mathbf{x}) = \frac{1}{n-2} \sum_{i=3}^{n} \left(\frac{x_i - 0.5}{0.5}\right)^2. \end{aligned} \tag{23}$$

Here, each variable x_i lies in $[0, 1]$. The modification makes the Pareto-optimal front a convex spherical surface. Figure 23 shows the solutions obtained with a single processor NSGA-II using

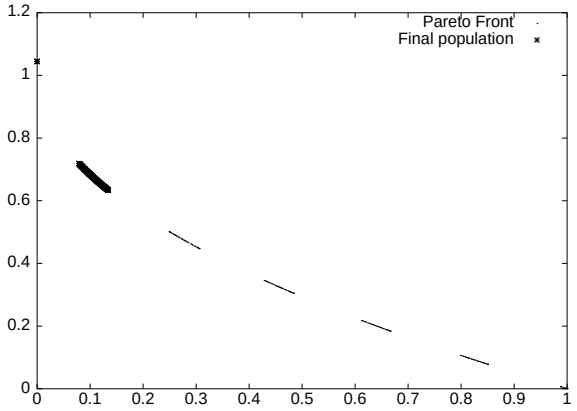

Fig. 21. Obtained non-dominated solution with single-processor NSGA-II on CTP7 problem.

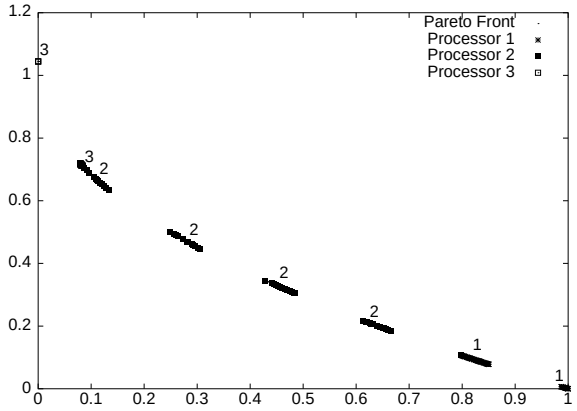

Fig. 22. Obtained non-dominated solution with 3-processor Parallel NSGA-II on CTP7 problem.

a population of size 300. Other NSGA-II parameters are the same as before, except that the crowding operator of NSGA-II is replaced with a *clustering* operator similar to that used in the SPEA [9]. Although this replacement increases the computational time by almost an order of magnitude, the obtained diversity in solutions is better. Figure 24 shows a typical distribution of solutions obtained by a three-processor clustered NSGA-II. The following transformation matrices are used:

$$\begin{bmatrix} 1 & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ 0 & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} \end{bmatrix}, \quad \begin{bmatrix} \frac{1}{\sqrt{2}} & 0 & 0 \\ -\frac{1}{\sqrt{2}} & 1 & -\frac{1}{\sqrt{2}} \\ 0 & 0 & \frac{1}{\sqrt{2}} \end{bmatrix}, \quad \begin{bmatrix} \frac{1}{\sqrt{2}} & 0 & 0 \\ 0 & \frac{1}{\sqrt{2}} & 0 \\ -\frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 1 \end{bmatrix}.$$

In each processor, 100 population members are used and 30 solutions are migrated after every 10 generations. In both cases, NSGA-IIs are run for 200 generations so that solutions converge very close to the Pareto-optimal front. The figure shows that each processor finds an adequate number of well-distributed set of solutions in its own designated region in the Pareto-optimal front. The computational time needed for the three-processor case is about 25 times lesser than that required with the single processor.

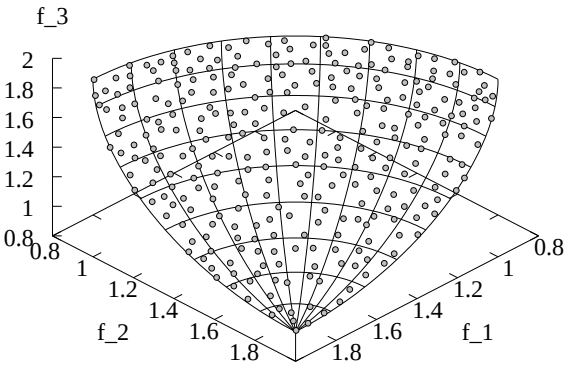

Fig. 23. Obtained solutions with a single processor for the modified DTLZ2.

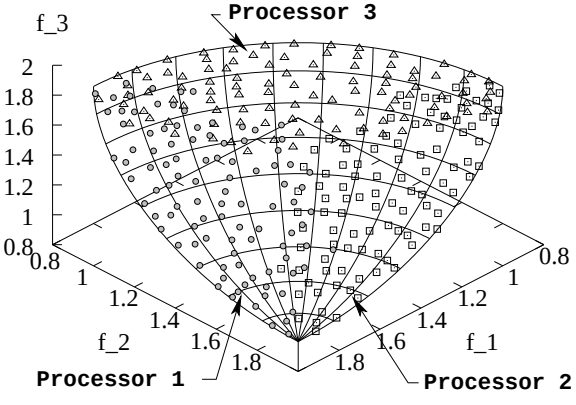

Fig. 24. Obtained solutions with three processors for the modified DTLZ2.

To investigate further, we have applied the clustered NSGA-II with six processors with an allocation plan shown in Figure 13. Figure 25 shows the agglomeration of solutions found in all six processors. Once again, a migration plan with 30 solutions migrating after every 10 generations is used. Each processor finds a well-distributed set of solutions in a particular region on the Pareto-optimal front.

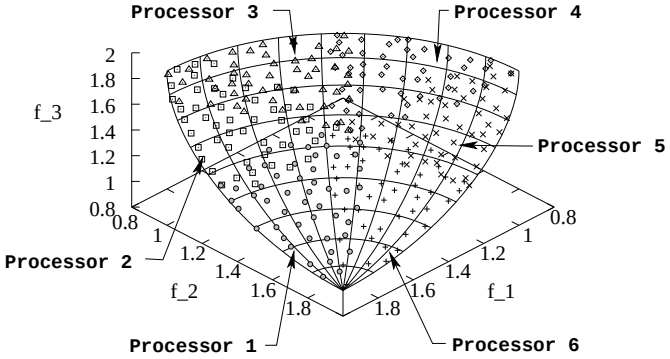

Fig. 25. Obtained solutions with six processors for the modified DTLZ2.

DTLZ4 test problem Finally, we apply the proposed technique to a modified DTLZ4 test problem having $n = 12$ variables:

$$\begin{aligned} &\text{Minimize } f_1(\mathbf{x}) = 2 - (1 - g(\mathbf{x})) \cos(x_1^\alpha \pi/2) \cos(x_2^\alpha \pi/2), \\ &\text{Minimize } f_2(\mathbf{x}) = 2 - (1 - g(\mathbf{x})) \cos(x_1^\alpha \pi/2) \sin(x_2^\alpha \pi/2), \\ &\text{Minimize } f_2(\mathbf{x}) = 2 - (1 - g(\mathbf{x})) \sin(x_1^\alpha \pi/2), \\ &\quad \text{where } g(\mathbf{x}) = \frac{1}{n-2} \sum_{i=3}^{n} \left(\frac{x_i - 0.5}{0.5} \right)^2. \end{aligned} \tag{24}$$

Here too, each x_i is initialized in $[0, 1]$ and $\alpha = 100$ is used. The reason for choosing this function is that an earlier study [6] has experienced difficulties in maintaining a diverse set of solutions in the original DTLZ4 problem due to the excessive density of solutions near $x_1 = 0$ and $x_2 = 0$ values. That study also experienced identical difficulties with SPEA (which uses the clustering approach for maintaining diversity). We have also observed a similar difficulty here with the clustered NSGA-II. However, Figure 26 shows one case (with a particular initial population) in which a good distribution is observed. All parameters used in this study are the same as that used in DTLZ2. Interestingly, the parallel clustered NSGA-II with three processors has always found good distribution of solutions on this problem. Figure 27 shows a typical agglomeration of solutions from all three processors after 200 generations. Here, 30 solutions are migrated after every 10 generations. Each processor used 100 population members, while the single-processor NSGA-II used a population of size 300.

5 Conclusions and Extensions

The study of multi-objective EAs with multiple processors has received a luke-warm interest so far. In this paper, we have proposed a technique in which each processor has been assigned the task of finding only a particular portion of the Pareto-optimal region. A simple yet effective allocating plan has been suggested by using direction cosines of transformed coordinate systems. Since all

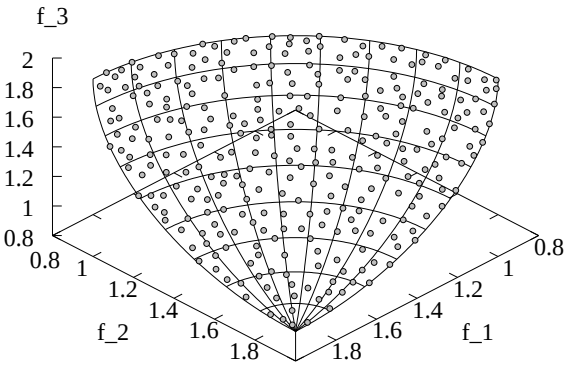

Fig. 26. Obtained solutions with a single processor for the modified DTLZ4.

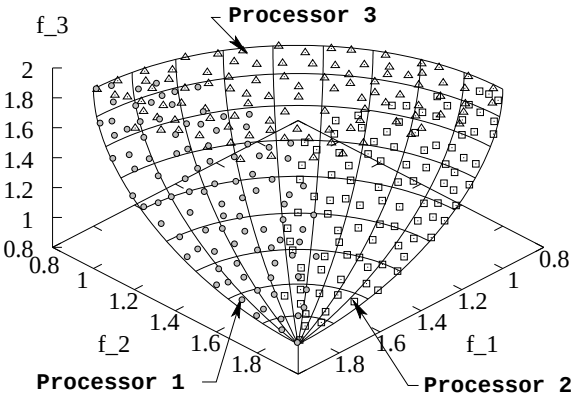

Fig. 27. Obtained solutions with three processors for the modified DTLZ4.

processors have to tackle the same complexities of the search space (starting from more or less the same region in the search space and overcoming similar hurdles to reach near the Pareto-optimal front), a migration policy of sending a few good non-dominated solutions from one processor to other processors has also been suggested. On a number of two and three objective test problems, the efficacy of the proposed technique has been demonstrated using NSGA-II. It has been observed that the use of multiple processors may be found beneficial in getting a widely distributed set of solutions with a super-linear computational speed-up.

The transformation technique proposed in this paper is illustrated for solving problems having a convex Pareto-optimal front. The approach can also be applied to problems having a non-convex Pareto-optimal front with appropriate choice of coordinate transformations. However, a different transformation technique can be tried with Tchebyshev metrics [3] and with other biasing techniques presented in Section 3. In an another island approach, an emphasis of the population members dissimilar to the migrated individuals may lead to formation of non-overlapping clusters of Pareto-optimal solutions. Nevertheless, since in most real-world applications convex Pareto-optimal fronts are encountered, the technique proposed in this paper should have a wide-scale applicability.

References

1. J. Branke, T. Kauβler, and H. Schmeck. Guiding multi-objective evolutionary algorithms towards interesting regions. Technical Report No. 399, Institute AIFB, University of Karlsruhe, Germany, 2000.
2. C. A. C. Coello, D. A. VanVeldhuizen, and G. Lamont. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Boston, MA: Kluwer Academic Publishers, 2002.
3. K. Deb. *Multi-objective optimization using evolutionary algorithms*. Chichester, UK: Wiley, 2001.
4. K. Deb. Multi-objective evolutionary algorithms: Introducing bias among Pareto-optimal solutions. In A. Ghosh and S. Tsutsui, editors, *Theory and Applications of Evolutionary Computation : Recent Trends*. London: Springer-Verlag, in press.
5. K. Deb, A. Pratap, and T. Meyarivan. Constrained test problems for multi-objective evolutionary optimization. In *Proceedings of the First International Conference on Evolutionary Multi-Criterion Optimization (EMO-01)*, pages 284–298, 2001.
6. K. Deb, L. Thiele, M. Laumanns, and E. Zitzler. Scalable multi-objective optimization test problems. In *Proceedings of the Congress on Evolutionary Computation (CEC-2002)*, pages 825–830, 2002.
7. I. C. Parmee, D. Cevtković, A. W. Watson, and C. R. Bonham. Multiobjective satisfaction within an interactive evolutionary design enviornment. *Evolutionary Computation Journal*, 8(2):197–222, 2000.
8. E. Zitzler, K. Deb, L. Thiele, C. A. C. Coello, and D. Corne, editors. *Evolutionary Multi-Criterion Optimization (Lecture Notes in Computer Science 1993)*. Heidelberg: Springer, 2001.

9. E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271, 1999.