# On the Performance of the Pareto Set Pursuing (PSP) Method for Mixed-Variable Multi-Objective Design Optimization

**Zeeshan Omer Khokhar**
MENRVA Research Group
zok@sfu.ca

**Hengameh Vahabzadeh**
Product Design and Optimization Lab
hva1@sfu.ca

**Amirreza Ziai**
MENRVA Research Group
amirreza_ziai@sfu.ca

**G. Gary Wang**[1]
Product Design and Optimization Lab
gary_wang@sfu.ca

**Carlo Menon**[1]
MENRVA Research Group
cmenon@sfu.ca

School of Engineering Science
Simon Fraser University
8888 University Drive
Burnaby, BC, V5A 1S6, Canada

*Practical design optimization problems require use of computationally expensive "black-box" functions. The Pareto Set Pursuing Method (PSP for solving multi-objective optimization problems with expensive black-box functions was developed originally for continuous variables. In this paper, modifications are made to allow solution of problems with mixed continuous-discrete variables. A performance comparison strategy for non-gradient-based multi-objective algorithms is discussed based on algorithm efficiency, robustness and closeness to the true Pareto front with a limited number of function evaluations. Results using several methods, along with the modified PSP, are given for a suite of benchmark problems and two engineering design ones. The modified PSP is found to be competitive when the total number of function evaluations is limited, but faces an increased computational challenge when the number of design variables increases.*

## 1 Introduction

Modern design problems often have to make trade-offs between conflicting objectives; these problems are known as multiobjective optimization (MOO) problems. The methods used for solving such MOO problems can be broadly categorized into two main classes. The first class of methods involves converting the multiple objectives into a single objective by use of implicit or explicit weights, preferences, utilities, or targets. As these methods require a priori selection of weights, preferences, or utilities [1, 2], finding a rigorous method for such a selection is a challenging task. Such a selection might not be able to adequately capture the decision makers' preferences. Also this class of methods is not able to find the Pareto points in non-convex regions in the performance space [3]. Physical programming and its extension to interactive multiobjective design are some more recent work in this class of methods [4-7].

The second class of methods is based on finding a set of discrete points as an approximation of the Pareto frontier. The most successful and widely used approaches in this category seem to be evolutionary algorithms [8-15]. Evolutionary algorithms (EAs) do not use the information about function and slope continuity and can be easily applied to optimization problems with mixed variables. Since these methods are essentially population based, they require evaluation of numerous solutions before converging to the best set of solutions. This fact prohibits using these methods for multi-objective optimization problems involving computationally expensive analysis [16]. A detailed survey of MOO methods can be found in Ref. [2].

[1]Corresponding authors

Many MOO problems in design involve expensive analysis and simulation processes such as finite element analysis (FEA) and computational fluid dynamics (CFD). The increasingly wide use of these tools brings new challenges to optimization. FEA and CFD simulations involve a large number of simultaneous equations and therefore are considered computationally expensive [17]. These processes are often treated as "black-box" functions. Only inputs and outputs are known for these functions, so traditional optimization methods cannot be applied to a black-box function. Although computers advance at a very high pace nowadays, these expensive processes also become more complex for greater accuracy [18].

In recent methods to solve MOO with black-box functions, each objective function is approximated or the Pareto front is approximated directly [7, 19-21]. A major issue with this approach is that the accuracy of the Pareto frontier obtained is dependent on the accuracy of the approximate models. Some methods have been proposed to combine EAs with approximation techniques so as to reduce the number of function evaluations [22, 23]. Such algorithms provide cost saving in terms of expensive function evaluations but still suffer the difficulty in finding the Pareto frontier points near the extreme points, which is inherited from EAs.

A new algorithm known as the Pareto set pursuing method (PSP) has been proposed in [17], which was specifically designed for expensive black box functions. It is expected that when there is no limit on the number of function evaluations, PSP will not be able to compete with other algorithms. This is because that PSP is specifically designed for expensive black box functions and requires extra computational time in fitting the metamodels. This computational time can easily be justified for expensive functions but may not be feasible for cheap multi-objective design problems. PSP method has been developed for continuous variables and hence will be referred as C-PSP. In this work an extension of this algorithm has been developed for solving mixed variable multiobjective design problems and will be referred as MV-PSP. It is to be noted that MV-PSP includes C-PSP and all the advantages of C-PSP are inherited in MV-PSP. Also a comprehensive comparison has been carried out between the state-of-the-art non-gradient based MOO algorithms and the MV-PSP. The authors tried to find mature metamodel-based algorithms for expensive black-box MOO problems to compare with MV-PSP, but our efforts did not avail. Before introducing the MV-PSP, an overview of C-PSP is provided in the next section. The modification for handling mixed variables is described in Section 3. Section 4 describes some of the performance metrics that are used to compare different methods. Section 5 introduces a suite of benchmark problems and presents the optimization results for these problems. Two engineering design examples are presented in Section 6 while Section 7 draws conclusions and provides future work.

## 2 Overview of Pareto set pursuing method for continuous variables (C-PSP)

The basic methodology followed for the C-PSP is based on direct sampling in order to approximate the Pareto frontier. The basic idea is to start from random samples in performance space and then to iteratively draw samples closer to the true Pareto frontier. If this sampling trend is continued, we can sample points right on or very close to the true Pareto frontier. The steps involved in C-PSP are shown in Figure 1.

### 2.1. Steps of C-PSP algorithm

*2.1.1 Step 1. Initial Random Sampling and Expensive Function Evaluations.* First some initial points are sampled to build an approximation model for each objective function. Both quadratic polynomial fitting (QPF) and linear radial basis function (RBF) are employed in PSP. These two methods are automatically alternated during the sampling procedure as described in [17]. For both QPF and RBF models, the number of initial random sample points is ($n$+1)($n$+2)/2, where $n$ denotes the number of variables. This is the minimum number of samples required to build a full quadratic approximation model. After random sampling, expensive black-box functions are called to evaluate these sample points. The number of the so-far evaluated sample points $np$ at the end of the first step thus equals to ($n$+1)($n$+2)/2.

*2.1.2 Step 2. Fitness Computation and Current Frontier Points Identification.* The fitness value of a given set of design points is defined as [8].

$$G_i = [1 - \max_{j \neq i}(\min(f_{s1}^i - f_{s1}^j, f_{s2}^i - f_{s2}^j, \cdots, f_{sm}^i - f_{sm}^j))]^l \quad (1)$$

Where, $i$ and $j$ are two designs in a given set, $G_i$ denotes the fitness value of the $i$th design; $f_{sk}^i$ is the scaled $k$th objective function value of the $i$th design, $k$ = 1,…, $m$; and $l$ is called the frontier exponent and is taken as a constant 1. The *max* operator in Eq. (1) is over all other designs $j \neq i$ in the set and the *min* is over all the objectives. The objectives $f_{sk}^i$ and $f_{sk}^j$ in Eq. (1) are scaled to a range [0, 1]. Using this fitness function the fitness value of each evaluated initial sample is calculated; and based on this fitness value current non-dominated set of points are identified. The mean fitness value of the currently identified points is also calculated for checking the convergence criterion. These currently identified frontier points are updated as the procedure progresses.

*2.1.3 Step 3. Objective Functions' Fitting and Sampling on Single Objective Functions.* In this step a large number of "cheap" points are generated independently from approximation models of each objective function individually. First the $np$ evaluated function values from Step 1 are used to build the QPF or RBF model for the objective function. Both random and discriminative sampling, using sampling guidance function, can now be used to get the desired large number of "cheap" points. A sampling guidance function for the $i$th objective function can be constructed as, $\hat{z}_i(x) = c_0 - \hat{f}_i(x)$, $c_0 \geq \hat{f}(x)$, where $\hat{z}_i(x)$ is the sampling guidance function, $c_0$ is a constant and $\hat{f}(x)$ is an approximation model of the original expensive function. This sampling guidance function has the tendency to generate more sample points in the region where the value of $f_i$ is minimum [24]. An equal number of sample points are drawn independently according to each objective function's sampling guidance function. If the point leading to min $f_i$ is identified, this point will be at one of the vertices of the Pareto frontier in the performance space and is usually called an extreme point. This can overcome the difficulties of GA-based algorithms in identifying extreme points.

*2.1.4 Step 4. Combining the Sample Points From Last Step With Current Frontier Points; Frontier Points Identification From the Combined Point Set.* In this step preparation for sampling new frontier points is carried out. First the current frontier points, which are the expensive points, are combined with sample points from step 3, which are cheap points, and the fitness value is recalculated for all the points in the combined set. For the current frontier points, their real function values are used in the fitness computation and for the sample points drawn from step 3 their respective $\hat{f}_i(x)$ function values are used instead. Sample points drawn from Step 3 enrich the information for the construction of a sampling guidance function for the next step, though their function values are only a prediction from the approximation model. After computing the fitness values for the combined set, points having value larger than 1 will be used for sampling in the next step. Other points are likely non-frontier points and are discarded.

*2.1.5 Step 5. Sampling Frontier Points.* The frontier points obtained in Step 4 are the "best" points among all of the existing points in terms of the possibility of becoming Pareto set points. For the frontier points obtained in Step 4 whose fitness value is larger than or equal to 1, another sampling guidance function is defined as $\hat{z}(x) = G_i - 1 = [1 - \max_{j \neq i}(\min(f_{s1}^i - f_{s1}^j, f_{s2}^i - f_{s2}^j, \cdots, f_{sm}^i - f_{sm}^j))]^l - 1$. This function samples the points in the region which has higher fitness value and thus represents our goal of sampling towards the Pareto frontier. A sample guided by the above function is drawn from the points obtained from Step 4. The number of samples drawn depends on the ratio of obtained sample points to the number of current frontier points in the last iteration. If there is a sample point that has already been evaluated then it is discarded to avoid re-evaluation.

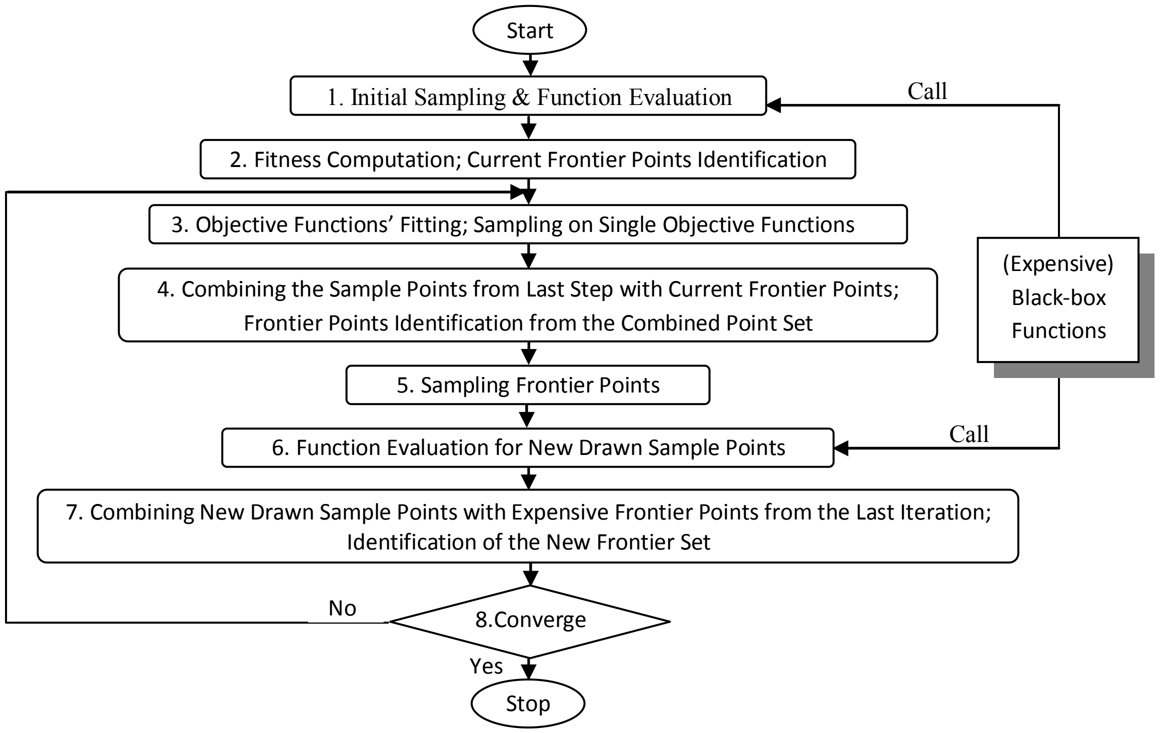

Start
1. Initial Sampling & Function Evaluation
Call
2. Fitness Computation; Current Frontier Points Identification
3. Objective Functions' Fitting; Sampling on Single Objective Functions
4. Combining the Sample Points from Last Step with Current Frontier Points; Frontier Points Identification from the Combined Point Set
(Expensive) Black-box Functions
5. Sampling Frontier Points
Call
6. Function Evaluation for New Drawn Sample Points
7. Combining New Drawn Sample Points with Expensive Frontier Points from the Last Iteration; Identification of the New Frontier Set
No
8.Converge
Yes
Stop

**Figure 1. Flowchart of the C-PSP approach**

*2.1.6 Step 6. Function Evaluation for New Sample Points*. In this step, the points in the new sample set obtained from Step 5 are evaluated by calling expensive black-box functions.

*2.1.7 Step 7. Combining New Sample Points with Expensive Frontier Points from the Last Iteration and Identification of the New Frontier Set.* At this step, the newly evaluated sample points are combined with previously evaluated expensive frontier points. Fitness values of this combined point set are then calculated and the frontier points are identified as the final frontier points of this iteration.

*2.1.8 Step 8. Checking Convergence.* If the convergence criteria are met, the procedure terminates; otherwise, back to Step 3. Two convergence criteria are applied here. The first one measures the progress of the iteration so that the difference between frontier points after two consecutive iterations is sufficiently small. The first criterion is met if 95% of the frontier points are the same as the previous iteration. The second criterion measures the closeness and distribution of frontier points on the Pareto frontier. If they are closely distributed then the fitness value will be very close to 1.

The C-PSP has been tested with a limited number of functions and has demonstrated high efficiency. It is however limited only to continuous variables and its performance needs to be benchmarked against other available multiobjective optimization methods for practical use.

## 3 Mixed Variable Version of Pareto Set Pursuing Method (MV-PSP)

Most of the real world design problems contain variables that are discrete or integers. The discrete variables mostly result from limited and standardized commercially available raw materials or off-the-shelf components, such as the thickness of a structural member or the size of a screw. Other discrete variables include, for example, the number of holes in a structure, number of turns in a coil, number of teeth in a gear, the type of material, and so on. Hence, a practical MOO method should be able to handle mixed variables.

The issue with C-PSP method with discrete variables arises in the random sampling stage as the samples generated may not belong to the desired set of values. This work extends C-PSP to MV-PSP for MOO problems of expensive black-box functions, involving all continuous variables, all discrete variables, or a mixture of both continuous and discrete variables.

In MV-PSP, each design variable can be defined as either discrete or continuous. Continuous design variables are specified within their upper and lower bounds:

$$x_r^l \le x_{c,r} \le x_r^u \tag{2}$$

Whereas discrete design variables are a finite set of values:

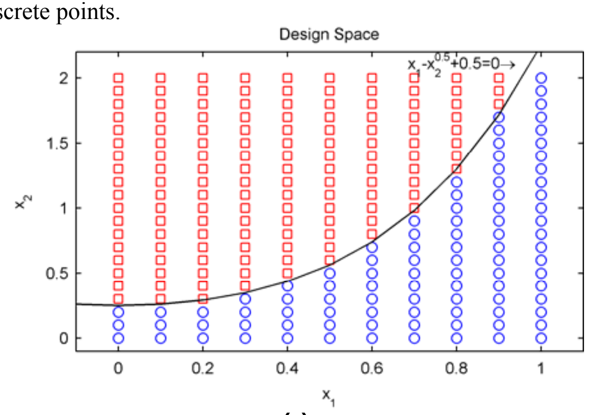
$$x_{D,r} = \{x_{D,r_1}, x_{D,r_2}, \ldots, x_{D,r_m}\} \tag{3}$$

Where $m$ is the number of elements within the design variable set, $r = 1,2,\ldots,n$ is the indicator of variable and $n$ is the total number of design variables.

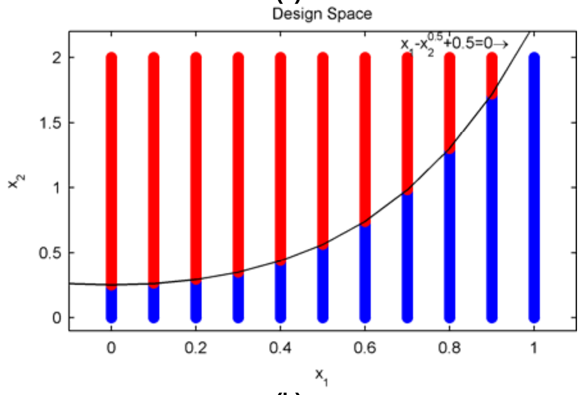
Feasible design space is defined as the set of all design points that satisfies the constraints. Obviously the feasible design space for continuous variables cannot be enumerated as there exists an infinite number of values for each continuous design variable. On the other hand, for discrete variables, the feasible design space is a finite set and can be enumerated. For instance, for the discrete variables and inequality constraint described below, the squares in Figure 2(a) represent the feasible design points and circles are points that do not satisfy the inequality constraint.

$$x_{D,1} = \{0, 0.1, 0.2, ..., 1\}$$
$$x_{D,2} = \{0, 0.1, 0.2, ..., 2\} \quad (4)$$
$$g_1(x) = x_{D,1}^2 - \sqrt{x_{D,2}} + 0.5 \le 0$$

Feasible design space for the cases where both discrete and continuous design variables are present cannot be enumerated. Assuming the second design variable of the above example is considered a continuous variable in the range of [0, 2] and the first design variable and the inequality constraint are left unchanged from the above example, the design space are the lines in Figure 2(b) with the lines above the curve become the feasible yet discontinuous design space. Given a discrete design space, the corresponding performance space and the Pareto frontier are also discrete points.



(a)



(b)

**Figure 2. Design space with all discrete variables (a), and mixed variables (b).**

In C-PSP, the first step is initial sampling and expensive function evaluation. In this step the minimum number of points necessary to construct a full quadratic metamodel of each objective function is $(n+1)(n+2)/2$. A problem arises with discrete variables where random sampling would most likely generate points that are not on the set. In order to overcome this problem, random integers in the range of each discrete variable's vector length (number of elements) are generated and the element corresponding to the random integer is picked as a random sample. For example for the following design variable, there exist five elements in the set.

$$x_{D,r} = \{x_{D,r_1}, x_{D,r_2}, ..., x_{D,r_5}\} = \{1,2,5,7,3,2\} \quad (5)$$

If we want to randomly pick three of the elements in the set, we first need to generate three random index numbers that are within the range of 1 and 5 (design variable vector length) and then pick elements according to these random indices. For the set in Eq. (5), if three indices are generated as 2, 3, and 5, then correspondingly $x$ = [2, 5, 32] are chosen as the discrete sample points. The sampled points are evaluated using the objective black-box functions. Fitness values are computed and the initial frontier points are identified as elaborated earlier.

After the initialization step, C-PSP then model the objective functions using the acquired sample points and then drawing a large number of "cheap" sample points from the metamodel. These points are called "cheap" because they are not evaluated by the expensive black-box function and are not computationally considered expensive or time-consuming. Using the same technique as described above, one can draw cheap points from the feasible design space that contains discrete design variables.

With some minor modifications, C-PSP is revised to take mixed variables for multiobjective optimization problems. The performance of the new MV-PSP is then tested for MOO problems with expensive black-box functions.

## 4 Quality Indicators for Performance Comparison

MOO performance measurement is completely different from single objective optimization (SOO). Unlike SOO that has a single optimal solution, MOO optimum solution is a set of solutions which makes it difficult to compare the results of two different methods. The quality of the solution obtained depends on a number of factors which includes the closeness of the points obtained to the Pareto frontier, the number of points obtained, and how well the points are distributed on the Pareto frontier. Refs. [25-29] gave a few quality metrics that are utilized in this work.

**4.1 Spread.** It is desired that the Pareto set be well spread. To quantify the spread, Euclidean distance between any two neighbor solutions in non-dominated solution set is calculated and then the average of these distances is obtained. Then the extreme points of the true Pareto frontier are found and Euclidean distance between these points and the boundary solutions of the obtained Pareto solution is calculated and is called as $d_f$ and $d_l$ . The non-uniformity in the distribution, Δ, is calculated using the equation below [26].

$$\Delta = \frac{d_f + d_l + \sum_{i=1}^{N-1} \left| d_i - \bar{d} \right|}{d_f + d_l + (N-1)\bar{d}} \quad (6)$$

The parameter $\bar{d}$ is the average of all distances $d_i$ , which are $dis(i,L_j)$ in crowding distance calculation, $i$=1,2,…($N$-1). With $N$ solutions, $d_i$'s are consecutive distances. Figure 3 indicates how this metric is calculated. A lower value of Δ indicates a better spread of Pareto optimal solution.

**4.2 Generational Distance.** This quality indicator is used for measuring distance between the elements in the obtained non-dominated points and the True Pareto frontier (TPF) (known a prior). It is defined as

$$GD = \frac{\sqrt{\sum_{i=1}^{k} r_i^2}}{k} \quad (7)$$

where k is the number of vectors in the set of non-dominated solutions found so far and $r_i$ is the Euclidean distance (measured in the objective space) between each of these solutions and the nearest member of the TPF. These distances are shown by solid lines between obtained solution and chosen points in Figure 4. A value of GD=0 indicates that all the generated elements are in the TPF [27].

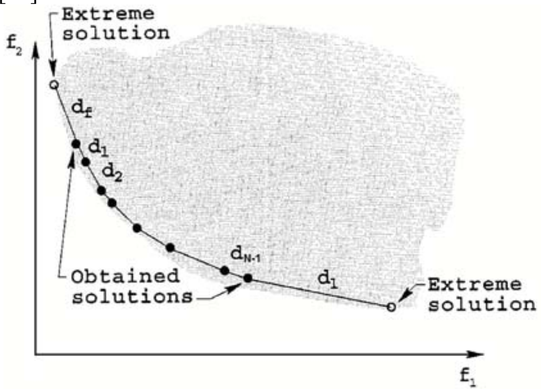


Figure 3. Illustration of variables used in calculating spread [26]

**4.3 Inverted Generational Distance.** This quality indicator is for measuring how far the elements are in the TPF from those in the obtained set of non-dominated points.

$$IGD = \frac{\sqrt{\sum_{i=1}^{k} q_i^2}}{k} \tag{8}$$

where $k$ is the number of vectors in the TPF set and $q_i$ is the Euclidean distance between each of these vectors and the nearest member of the set of the obtained non-dominated points. These distances include the solid as well as dotted lines between obtained solution and chosen points shown in Figure 4. *IGD=0* indicates that all of the generated elements are in the TPF and all the extension of the Pareto frontier are covered. Non-dominated sets should be normalized before calculating the distance measures [27].

It is to be noted that although Eqs. (7) and (8) have the same form, GD measures the distances from the obtained frontier to TPF, while IGD measures the distance from TPF to the obtained frontier. Assume two obtained frontiers of a given problem: one only has two points, and the other has the two points in the first frontier and many other non-dominated points. It is clear that from the multi-objective perspective that the second frontier is preferable to the first one. However, *GD* cannot differentiate the two frontiers because it only measure the distances from the obtained frontier to TPF. Meanwhile *IGD* can tell that the second one is better because the distances are measured from TPF to the obtained frontier.

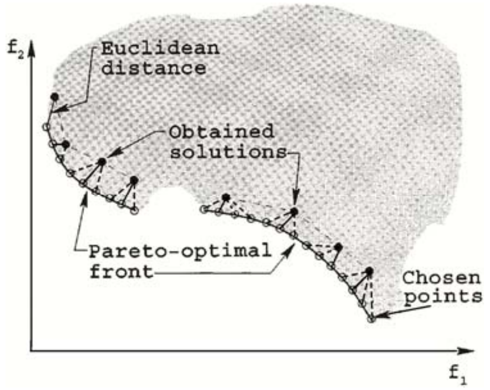


Figure 4. Illustration of Euclidean distance for calculating generational distance and inverted generational distance [26]

**4.4 Hypervolume.** This quality indicator calculates the volume, in the objective space, covered by members of the non-dominated set of solutions. In other words, for each solution *i*, a hypercube $v_i$ is constructed such that the solution *i* and a reference point *W* are its diagonal corners of the hypercube. The reference point can be simply constructed by finding a vector of worst objective function values. Then union of all such hypercubes are found and the hypervolume (*HV)* is calculated

$$HV = volume\left(\bigcup_{i=1}^{|Q|} v_i\right) \tag{9}$$

Algorithms that have larger values of *HV* are more desirable. Here also the normalized objective values should be used in calculation. This measure does not need TPF and focuses mostly on the “goodness” of the solutions rather than spread.

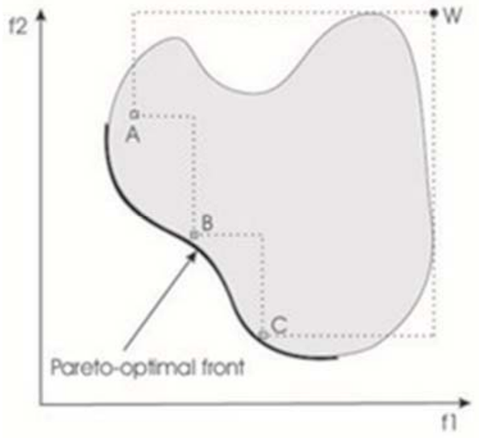


Figure 5. Illustration of calculation of vi [28]

**4.5 Generalized Spread.** The previously presented metric, spread, calculates the distance between two consecutive solutions, which only works for 2-objective problems. A generalized metric was introduced in [29], which calculates the distance from a point to its nearest neighbor from Equation (10a)

$$\Delta(PF,TPF) = \frac{\sum_{i=1}^{m} d(e_i,PF) + \sum_{X\in TPF}\left|d(X,PF) - \bar{d}\right|}{\sum_{i-1}^{m} d(e_i,PF) + |TPF|\bar{d}} \tag{10a}$$

where PF is the current optimal Pareto frontier, TPF is the true Pareto frontier and $\{e_1,\ldots,e_m\}$ are *m* extreme solutions in TPF, and,

$$d(X,S) = \min_{Y\in PF, Y\neq X} \left\|F(X) - F(Y)\right\|^2,$$
$$\bar{d} = \frac{1}{|TPF|}\sum_{X\in TPF} d(X,PF) \tag{10b}$$

**4.6 Percentage of Frontier Points.** When comparing expensive objective functions, it is desirable that the Pareto frontier solution converges to the true Pareto frontier with minimum number of function evaluations so that we can save computational time. The percentage of Pareto frontier points in the total number of function evaluation should be maximized for a computationally expensive objective function.

**Table I. Test bed functions**

| Problem | $n$ | Variable bounds | Objective function | Characteristics |
|---|---|---|---|---|
| SCH [30] | 1 | $[-10^3, 10^3]$ | $f_1 = x^2$ <br> $f_2 = (x-2)^2$ | Convex; simple |
| FON [31] | 3 | [-4,4] | $f_1 = 1 - \exp(-\sum_{i=1}^{3}(x_i - \frac{1}{\sqrt{3}})^2)$ <br> $f_1 = 1 - \exp(-\sum_{i=1}^{3}(x_i + \frac{1}{\sqrt{3}})^2)$ | Non-convex |
| KUR [32] | 3 | [-5,5] First two variables discretized with 0.1 step size | $f_1 = \sum_{i=1}^{n-1}(-10\exp(-0.2\sqrt{x_i{}^2 + x_{i+1}{}^2}))$ <br> $f_2 = \sum_{i=1}^{n}(\lvert x_i\rvert^{0.8} + 5\sin x_i{}^3)$ | Multi-modal function in one objective; pair-wise variable interactions in the other; Pareto frontier not connected with both concave and convex regions |
| ZDT6 [33] | 10 | [0,1] | $f_1 = 1 - \exp(-4x_1 \sin^6(6\pi x_1))$ <br> $f_2 = g(x)[1 - (f_1(x)/g(x))^2]$ <br> $g(x) = 1 + 9[(\sum_{i=2}^{n} x_i)/(n-1)]^{0.25}$ | Unimodal; non-uniformly distributed objective space |
| DTLZ1 [34] | 7 | [0,1] | $f_1 = \frac{1}{2}x_1x_2(1+g(x))$ <br> $f_2 = \frac{1}{2}x_1(1-x_2)(1+g(x))$ <br> $f_3 = \frac{1}{2}(1-x_1)(1+g(x))$ <br> $g(x) = 100(\lvert x_3\rvert + \sum_{i=3}^{n}(x_i - 0.5)^2 - \cos(20\pi(x_i - 0.5)))$ <br> $x_i \in [0,1] \qquad i = 1,2,\ldots\ldots,n$ | Hyperplane Pareto frontier with many local Pareto frontiers due to the constraint |

## 5 Performance Comparison using Benchmark Problems

In order to calculate the performance measurements described above, it is desirable to have test problems for which the TPF is known. There have been numerous studies in defining test problems for comparison of multiobjective optimization algorithms. We will consider five test problems for our comparison, each with different characteristics. These are SCH, KUR, FON, ZDT6 & DTLZ1 (See Table I). The comparison of the PSP method is carried out with six different EAs, namely Abyss [35], CellDE [36], FastPGA [37], NSGAII [26], OMOPSO [38] & SPEA2 [39]. In order to compare the multiobjective optimization algorithms, each algorithm is allowed to run for the test problems for a constant number of function evaluations. The quality indicators are calculated for each algorithm run. This procedure is repeated for thirty runs and the average and standard deviation of the quality indicators are recorded for each algorithm. The codes for the different algorithms and calculation of quality indicators are available in jmetal [28]. The default values of crossover and mutation probabilities are used, which is 0.9 for crossover probability and $1/n$ for mutation probability, where $n$ is the number of variables in the test problem. Some of the parameters like the initial population size and archive size had to be changed in order for the algorithms to work with a maximum of 50 function evaluations. For the PSP algorithm the convergence criterion had to be bypassed to allow it to run for a constant number of function evaluations.

**5.1. SCH (*n*=1)**. Since SCH is a low dimensional problem, each algorithm is allowed to run first for 50 function evaluations then for 100 and finally for 200. Then this process is repeated for 30 independent runs. Appendix Table 1 shows the average and standard deviation of different quality indicators of 30 runs for SCH test problem for different algorithms and 50 function evaluations. Abyss was not able to produce any Pareto points and therefore is not included in the table. Figure 6 shows the Pareto points obtained with reference to the true Pareto frontier graphically, for 50 function evaluations using results from one of the 30 runs. For 50 function evaluations, none of the other algorithms apart from PSP was able to produce any Pareto points close to the TPF. The points obtained from other algorithms were so far apart from TPF that they are not visible in Figure 6. For 200 function evaluations every algorithm was able to generate Pareto points right on TPF but the performance of PSP was much better, generating more than 97% Pareto points from the total number of function evaluations.

**5.2. FON (*n*=3)**. Appendix Table 2 summarize the results for 30 simulations for different algorithms with 100 function evaluations. Figure 7 provides a graphical visualization of the Pareto points obtained for 200 function evaluations. As one can see from the Table II, PSP outperforms the others with 50 function evaluations. When the allowed number of function evaluations increases, other algorithms gradually catch up, which leaves PSP a ranking of 3 and 5 respectively for 100 and 200 function evaluation cases.

**5.3. KUR (*n*=3)**. To show the performance of MV-PSP with mixed variables, first two variables of the KUR problem were discretized with a step size of 0.1. Other evolutionary algorithms do not have built in capability to handle mixed variables and so for comparisons the decision variables were discretized after running the optimization algorithms and function values were calculated for the new decision variables. Appendix Table 3 lists the results for the KUR test problem for 100 function evaluations, for which the graphical representation is shown in Figure 8. The performance of PSP in this case is also far better than the other evolutionary algorithms. It was seen that PSP generates points near all the three segments of the TPF, where many of the other algorithms were not able to do so.

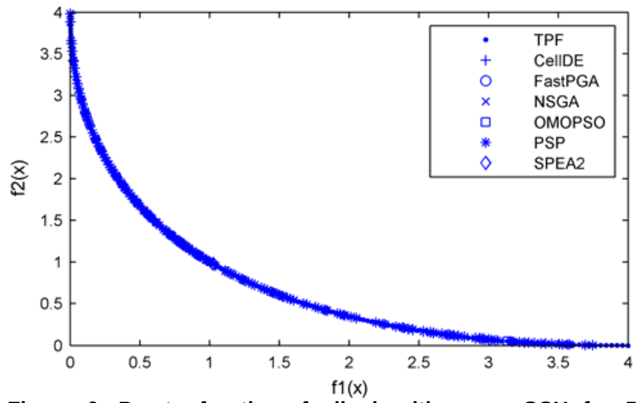


**Figure 6. Pareto frontier of all algorithms on SCH for 50 function evaluations**

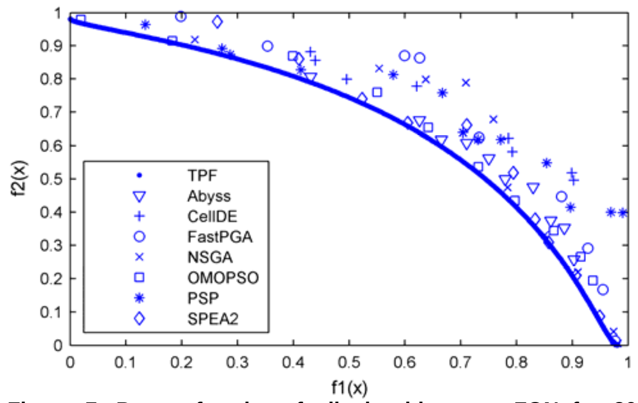


**Figure 7. Pareto frontier of all algorithms on FON for 200 function evaluations**

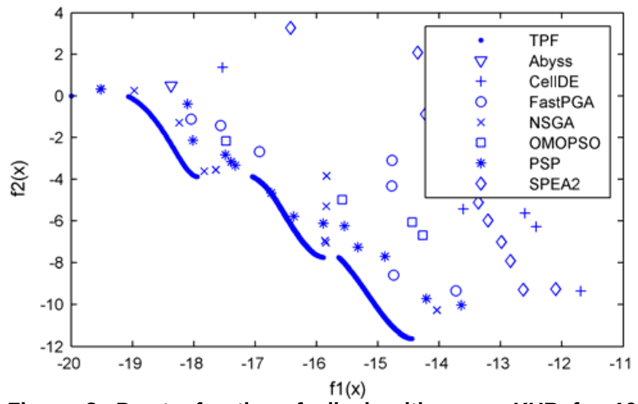


**Figure 8. Pareto frontier of all algorithms on KUR for 100 function evaluations**

**5.4. ZDT6 (*n*=10)**. Since ZDT6 has 10 variables, we use 200, 500, and 1000 function evaluations as the stopping criteria to allow more computation.

From Figure 9, it is clear that none of the algorithms were able to capture the TPF, mostly due to the vast search space caused by the high dimension and the limited number of function evaluations. From Table II, PSP ranks 4, 2, 4, respectively for the three cases.

**5.5. DTLZ1 (*n*=7)**. Since this problem has relative high number of variables, the algorithms for this problem are also run for 200, 500 and 1000 function evaluations. The test results for 500 function evaluations are listed in Appendix Table 5. As the number of function evaluations increases, each algorithm progresses towards the Pareto frontier. PSP ranks number 1 for all three cases. The graphical illustration is in Figure 10.

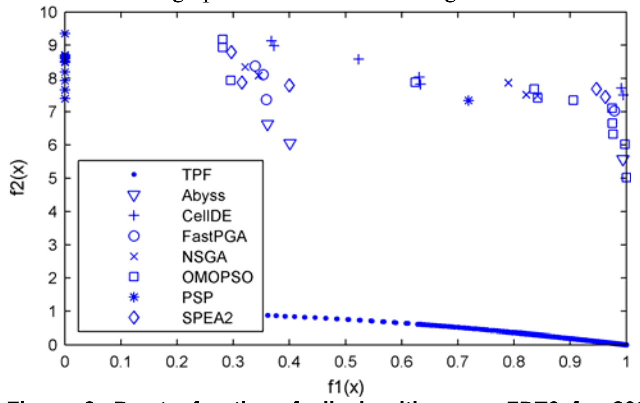


**Figure 9. Pareto frontier of all algorithms on ZDT6 for 200 function evaluations**

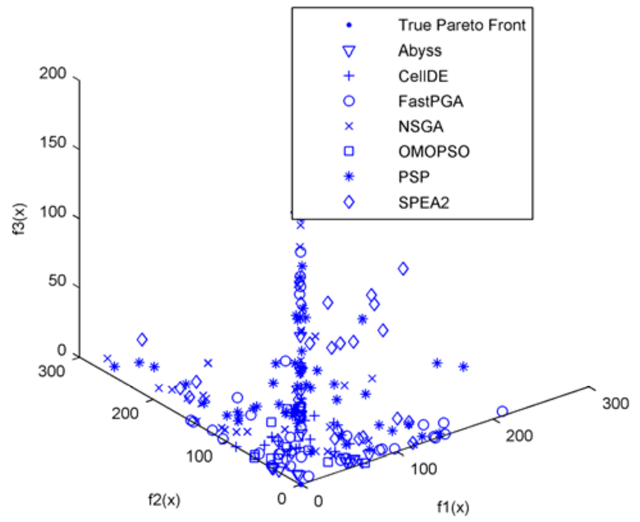


**Figure 10. Pareto frontier of all algorithms on DTLZ1 for 500 function evaluations**

**Table II. Ranking of PSP in relation to other six state-of-the-art MOO algorithms for the test suite**

| Problem | 200 func. eval. for ZDT6 and DTLZ1; 50 for others | 500 func. eval. for ZDT6 and DTLZ1; 100 for others | 1000 func. eval. for ZDT6 and DTLZ1; 200 for others |
|---|---|---|---|
| SCH (*n*=1) | 1 | 1 | 1 |
| FON (*n*=3) | 1 | 3 | 5 |
| KUR (*n*=3) | 1 | 1 | 1 |
| ZDT6 (*n*=10) | 4 | 2 | 4 |
| DTLZ1 (*n*=7) | 1 | 1 | 1 |

From the test results in Appendix and the summary in Table II, with the limited number of function evaluations (in relation to the number of variables), PSP yields overall better solutions than other algorithms for the test problem suite, which has a range of different characteristics such as segmented frontier, non-convex regions, multimodal, and so on. The performance of all algorithms deteriorates for high dimensional problems (*n*=10) in this case, or the total number of function evaluations is too few to yield meaningful results. The advantages of PSP on the high dimensional problem, ZDT6, are also questionable. PSP, does, however outperforms others on DTLZ1, an $n$=7 problem with clear advantages. Depending on the performance space characteristics, the number of objective functions, and the number of variables, it is, however, difficult to draw a conclusion on with how many function evaluations PSP will outperform the others. It is relatively safe to say for problems of lower dimensionality with 2 to 3 objective functions, PSP is more efficient than other algorithms with a few dozens or hundreds of function evaluations.

## 6. Application Examples

To test the performance of PSP on engineering problems with mixed variables, two problems are chosen from the literature.

**6.1. Welded Beam Design Problem.** The welded beam design problem is a four variable problem, $\mathbf{x} = (h, l, t, b)$, with four non-linear constraints. It has two objectives, one is to minimize the cost of fabrication and the other is to minimize the end deflection of the welded beam [40]:

$$\begin{aligned}
&\text{Minimize } f_1(\vec{x}) = 1.10471\, h^2 l + 0.04811\, tb\,(14 + l),\\
&\text{Minimize } f_2(\vec{x}) = \frac{2.1952}{t^3 b},\\
&\text{Subject to } g_1(\vec{x}) = 13{,}600 - \tau(\vec{x}) \ge 0\\
&\qquad g_2(\vec{x}) = 30{,}000 - \sigma(\vec{x}) \ge 0\\
&\qquad g_3(\vec{x}) = b - h \ge 0\\
&\qquad g_4(\vec{x}) = P_c(\vec{x}) - 6{,}000 \ge 0\\
&\qquad 0.125 \le h, b \le 5.0\\
&\qquad 0.1 \le l, t \le 10.0
\end{aligned} \tag{11}$$

The stress and buckling terms are non-linear function of design variables and are given below.

$$\begin{aligned}
&\tau(\vec{x}) = \sqrt{(\tau')^2 + (\tau'')^2 + \frac{l\tau'\tau''}{\sqrt{0.25(l^2 + (h+t)^2)}}},\\
&\tau' = \frac{6{,}000}{\sqrt{2}hl},\\
&\tau'' = \frac{6{,}000(14 + 0.5l)\sqrt{0.25(l^2 + (h+t)^2)}}{2\{0.707hl(l^2/12 + 0.25(h+t)^2)\}},\\
&\sigma(\vec{x}) = \frac{504{,}000}{t^2 b},\\
&P_c(\vec{x}) = 64{,}746.022(1 - 0.0282346\ t)\, tb^3
\end{aligned} \tag{12}$$

Though all the variables are dimensions and thus continuous, we discretized all the variables with step of 0.1, and applied MV-PSP to solve the problem. We also solved the problem using other evolutionary algorithms for comparison. As these algorithms cannot handle discrete variables, we discretized the design variables after running the optimization algorithms and used the new values to recalculate the objective functions. We performed 30 independent runs and obtain the average numbers and recorded them in Table III.

Because the true Pareto frontier for the problem is not available, we cannot examine the accuracy of this method by means of the quality indicator indices that we used for the test suite. We can observe the solutions graphically as depicted in Figure 11. All the algorithms produced results very close to each other with MV-PSP producing slightly worse results than the other in terms of optimum design but the advantage of using MV-PSP is apparent from Table III which shows that it only takes an average of 228 function evaluations while other algorithms require more than 25000 function evaluations.

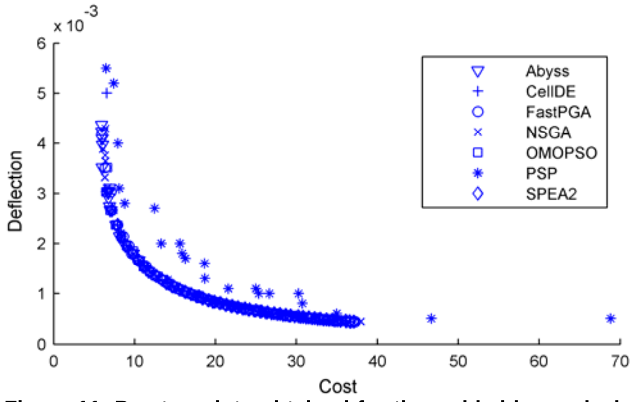


**Figure 11. Pareto points obtained for the welded beam design problem**

**Table III. Optimization Results for Welded Beam Design Problem with 30 independent runs, 1st column specifies average while 2nd column specifies standard deviation.**

| Algorithm | Number of Function Evaluations | | Number of Pareto Points | | Percentage of Pareto Points | |
|---|---|---|---|---|---|---|
| **PSP** | 227.97 | 95.5 | 19.97 | 2.78 | 10.68 | 5.51 |
| **Abyss** | 25018.1 | 13.28 | 55.47 | 6.24 | 0.22 | 0.02 |
| **CellDE** | 25000.0 | 0.00 | 45.57 | 0.90 | 0.18 | 0.00 |
| **FastPGA** | 25000.0 | 0.00 | 49.90 | 4.40 | 0.20 | 0.02 |
| **NSGAII** | 25000.0 | 0.00 | 47.90 | 3.08 | 0.19 | 0.01 |
| **OMOPSO** | 25100.0 | 0.00 | 47.90 | 1.97 | 0.19 | 0.01 |
| **SPEA2** | 25000.0 | 0.00 | 45.50 | 1.01 | 0.18 | 0.00 |

**6.2. Spring Design Problem.** The second engineering problem is the spring design problem consisting of two discrete variables and one continuous variable. The Pareto optimal frontier has a discrete set of solutions as a result of the discrete nature of the problem. The objectives are to minimize the volume of spring and minimize the stress developed by applying a load. Variables are diameter of the wire ($d$), diameter of the spring ($D$) and the number of turns ($N$). Denoting the variable vector x = ($x_1$, $x_2$, $x_3$) = ($N$, $d$, $D$), formulation of this problem with two objective and eight constraints is as follows [40]:

$$\begin{aligned}
&\text{Minimize } f_1(\vec{x}) = 0.25\pi^2 x_2^2 x_3 (x_1 + 2),\\
&\text{Minimize } f_2(\vec{x}) = \frac{8KP_{\max} x_3}{\pi x_3^2},\\
&\text{Subject to } g_1(\vec{x}) = l_{\max} - \frac{P_{\max}}{k} - 1.05(x_1 + 2) \ge 0,\\
&\qquad g_2(\vec{x}) = x_2 - d_{\min} \ge 0,\\
&\qquad g_3(\vec{x}) = D_{\max} - (x_2 + x_3) \ge 0,\\
&\qquad g_4(\vec{x}) = C - 3 \ge 0,\\
&\qquad g_5(\vec{x}) = \delta_{pm} - \delta_p \ge 0,\\
&\qquad g_6(\vec{x}) = \frac{P_{\max} - P}{k} - \delta_w \ge 0,
\end{aligned} \tag{13}$$

$$g_7(\vec{x}) = S - \frac{8KP_{\max} x_3}{\pi x_2^3} \geq 0,$$

$$g_8(\vec{x}) = V_{\max} - 0.25\pi^2 x_2^2 x_3 (x_1 + 2) \geq 0$$

$x_1$ is an integer; $x_2$ is a discrete variable; $x_3$ is a continuous variable.

The parameters used are as follows:

$$K = \frac{4C-1}{4C-4} + \frac{0.615x_2}{x_3}, \quad k = \frac{Gx_2^4}{8x_1x_3^3}, \quad \delta_p = \frac{P}{k}, \quad P = 300\text{ lb},$$

$$D_{\max} = 3\text{ in}, \quad P_{\max} = 1{,}000\text{ lb}, \quad \delta_w = 1.25\text{ in}, \quad \delta_{pm} = 6\text{ in},$$

$$S = 189\text{ ksi}, \quad d_{\min} = 0.2\text{ in}, \quad G = 11{,}500{,}000\text{ lb/in}^2,$$

$$V_{\max} = 30\text{ in}^3, \quad l_{\max} = 14\text{ in}, \quad C = x_3 / x_2.$$

The 42 discrete values of $d$ are given below:

$$\begin{pmatrix} 0.009 & 0.0095 & 0.0104 & 0.0118 & 0.0128 & 0.0132 & 0.014 \\ 0.015 & 0.0162 & 0.0173 & 0.018 & 0.02 & 0.023 & 0.025 \\ 0.028 & 0.032 & 0.035 & 0.041 & 0.047 & 0.054 & 0.063 \\ 0.072 & 0.08 & 0.092 & 0.105 & 0.12 & 0.135 & 0.148 \\ 0.162 & 0.177 & 0.192 & 0.207 & 0.225 & 0.244 & 0.263 \\ 0.283 & 0.307 & 0.331 & 0.362 & 0.394 & 0.4375 & 0.5 \end{pmatrix}$$

The obtained Pareto frontier is plotted in Figure 12, which shows that all algorithms generated solutions quite close to each other. Table IV summarizes the average results for 30 runs. One can see that with a modest total number of function evaluations, PSP generates a large percentage of points on the Pareto frontier, which present many alternative designs.

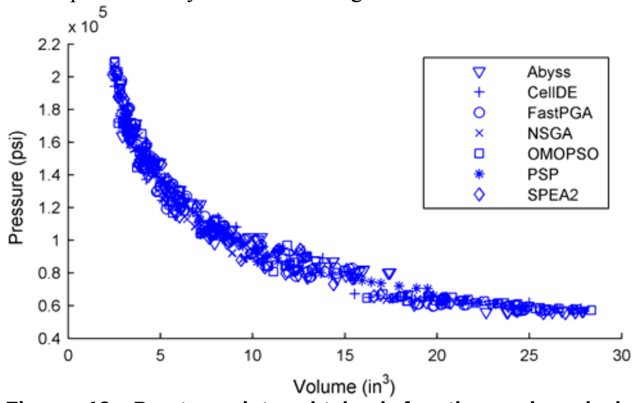


**Figure 12. Pareto points obtained for the spring design problem**

**Table IV. Optimization Results for Spring Design Problem with 30 independent runs, 1st column specifies average while 2nd column specifies standard deviation.**

| Algorithm | Average number of Function Evaluations | | Average number of Pareto Points | | Average Percentage of Pareto Points | |
|---|---|---|---|---|---|---|
| **PSP** | 147.10 | 43.80 | 49.13 | 5.42 | 35.21 | 7.16 |
| **Abyss** | 25016.7 | 12.07 | 90.30 | 3.24 | 0.36 | 0.01 |
| **CellDE** | 25000.0 | 0.00 | 98.37 | 1.25 | 0.39 | 0.00 |
| **FastPGA** | 25000.0 | 0.00 | 82.50 | 3.44 | 0.33 | 0.01 |
| **NSGAII** | 25000.0 | 0.00 | 78.97 | 3.69 | 0.32 | 0.01 |
| **OMOPSO** | 25100.0 | 0.00 | 101.0 | 0.00 | 0.40 | 0.00 |
| **SPEA2** | 25000.0 | 0.00 | 94.07 | 2.05 | 0.38 | 0.01 |

## 7. Conclusions

In this work, we have modified the continuous variable version of Pareto set pursuing (PSP) algorithm to handle mixed variables. Then the performance of PSP was thoroughly tested using a set of quality indicators with a well-selected benchmark test suite. Its performance was compared with the state-of-the-art multiobjective optimization algorithms. With limited function evaluations, PSP outperforms all the others for lower dimensional problems ($n$<8) with 2-3 objective functions. PSP offers no apparent advantage when the dimensionality increases to 10 or more.

In summary, for design problems with expensive function evaluations (the time for one function evaluation is at least one order of magnitude larger than the time for metamodeling), when the dimensionality is low ($n \leq 10$) with 2-3 objective functions, PSP is recommended for multiobjective optimization with limited budget. Future work will replace the metamodeling technique in PSP to accommodate larger dimensions.

## 8. Acknowledgements

This work was financially supported by the Natural Science and Engineering Research Council (NSERC) of Canada.

## 9. References

[1] Keeney, R. L. and Raifa, H., 1976, *Decisions with multiple objective: preferences and value trade-off,* John Wiley and Sons, New York.

[2] Marler, R. T. and Arora, J. S., 2004, "Survey of multi-objective optimization methods for engineering," *Structural and Multidisciplinary Optimization,* 26, pp. 369-395.

[3] Chen, W., Wiecek, M. M., and Zhang, J., 1999, "Quality utility -- a compromise programming approach to robust design," *ASME Transactions, Journal of Mechanical Design,* 121, pp. 179-187.

[4] Messac, A., 1996, "Physical programming: effective optimization for computational design," *AIAA Journal*, 34(1), pp. 149-158.

[5] Tappeta, R. V. and Renaud, J. E., 1999, "Interactive multiobjective optimization procedure," *AIAA Journal*, 37(7), pp. 881-889.

[6] Tappeta, R. V., Renaud, J. E., Messac, A., and Sundararaj, G., 2000, "Interactive physical programming: tradeoff analysis and decision making in multicriteria optimization," *AIAA Journal*, 38(5), pp. 917-926.

[7] Tappeta, R. V., and Renaud, J. E., 2001, "Interactive Multiobjective Optimization Design Strategy for Decision Based Design," *ASME Transactions, Journal of Mechanical Design*, 123, pp. 205-215.

[8] Schaumann, E. J., Balling, R. J., and Day, K., 1998, "Genetic algorithms with multiple objectives," *7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, St. Louis, MO, AIAA Vol. 3, Sept. 2-4, pp. 2114-2123, Paper No. AIAA-98-4974.

[9] Deb, K., 1999, "Evolutionary algorithms for multi-criterion optimization in engineering design," *Proceedings of Evolutionary Algorithms in Engineering & Computer Science*, Eurogen-99.

[10] Deb, K., Mohan, M. and Mishra, S., 2003, *A fast multi-objective evolutionary algorithm for finding well-spread pareto-optimal solutions*, Report No. 2003002, Indian Institute of Technology Kanpur, Kanpur.

[11] Srinivas, N. and Deb, K., 1995, "Multiobjective optimization using nondominated sorting in genetic algorithms," *Journal of Evolutionary Computation*, 2(3), pp. 221-248.

[12] Nain, P. K. S. and Deb, K., 2002, *A computationally effective multi-objective search and optimization technique using coarse-to-fine grain modeling*, Report No. 2002005, Indian Institute of Technology Kanpur, Kanpur.

[13] Luh, G. C., Chueh, C. H. and Liu, W. W., 2003, "MOIA: multi-objective immune algorithm," *Journal of Engineering Optimization*, 35(2), pp. 143-164.

[14] Deb, K., and Jain, S., 2003, "Multi-speed gearbox design using multi-objective evolutionary algorithms," *ASME Transactions, Journal of Mechanical Design*, 125, pp. 609-619.

[15] Saitou, K., and Cetin, O. L., 2004, "Decomposition-based assembly synthesis for structural modularity," *ASME Transactions, Journal of Mechanical Design*, 126, pp. 234-243.

[16] Isaacs, A., Ray, T., and Smith, W., 2009, "Multi-objective design optimization using multiple adaptive spatially distributed surrogates," *International Journal of Product Development*, 9(1/2/3), pp. 188-217.

[17] Shan, S., and Wang, G. G., 2005, "An efficient pareto set identification approach for multi-objective optimization on black-box functions," *ASME Transactions, Journal of Mechanical Design,* 127(5), pp. 866-874.

[18] Duan, X., Wang, G. G., Kang, K., Niu, Q., Naterer, G., Peng, Q., 2009, "Performance study of mode-pursuing sampling method," *Journal of Engineering Optimization*, 41(1), pp. 1-21.

[19] Wilson, B., Cappelleri, D. J., Simpson, T. W. and Frecker, M. I., 2001, "Efficient pareto frontier exploration using surrogate approximations," *Optimization and Engineering,* 2, pp. 31-50.

[20] Li, Y., Fadel, G. M. and Wiecek, M. M., 1998, "Approximating pareto curves using the hyper-ellipse," *7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, St. Louis, Paper No. AIAA-98-4961.

[21] Yang, B. S., Yeun, Y. S., and Ruy, W. S., 2003, "Managing approximation models in multiobjective optimization," *Structural and Multidisciplinary Optimization*, 24, pp. 141-156.

[22] Li, M., Li, G. and Azarm, S., 2008, "A kriging matamodel assisted multi-objective genetic algorithm for design optimization," *ASME Transactions, Journal of Mechanical Design*, 130(3), pp. 031401.

[23] Karakasis, M. K. and Giannakoglou, K. C., 2005, "Metamodel assisted multi-objective evolutionary optimization," *ECCOMAS Thematic Conference in Munich*, Sep. 12-14, 2005.

[24] Wang, L., Shan, S., and Wang, G. G., 2004, "Mode-pursuing sampling method for global optimization on expensive black-box functions," *Journal of Engineering Optimization,* 36(4), pp. 419-438.

[25] Wu, J., and Azarm, S., 2001, "Metrics for Quality Assessment of a Multiobjective Design Optimization Solution Set," *ASME Transactions, Journal of Mechanical Design*, Vol. 123, pp. 18--25, 2001.

[26] Deb, K., Pratap, A., Agarwal, S., Meyarivan, T., 2002, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, 6: 183-197.

[27] Hansen, S. R., and Vanderplaats, G. N., 1990, "Approximation method for configuration optimization of trusses," *AIAA Journal*, 28, pp. 161-168.

[28] Durillo, J. J., Nebro, A. J., Luna, F., Dorronsoro B., Alba, E., 2006, "{jMetal}: A Java Framework for Developing Multi-Objective Optimization Metaheuristics," *Departamento de Lenguajesy Ciencias dela Computaci'on, University of Malaga, E.T.S.I. Informatica, Campus de Teatinos,* ITI-2006-10, http://mallba10.lcc.uma.es/wiki/index.php/Tools, last access: Sept 17, 2009.

[29] Zhou A., Jin Y., Zhang, Q., Sendhof, B. and Tsang E., 2006, "Combining model-based and genetic-based offspring generation for multi-objective optimization using a convergence criterion," *2006 IEEE Congress on Evolutionary Computation*, pp. 3234-3241.

[30] Schaffer, J. D., 1985, "Multiple objective optimization with vector evaluated genetic algorithms," *Proceedings of the First International Conference on Genetic Algorithms and Their Applications: July 24-26, 1985 at the Carnegie-Mellon University, Pittsburgh,* pp. 93–100.

[31] Fonseca, C. M. and Fleming, P. J., 1998, "Multiobjective optimization and multiple constraint handling with evolutionary algorithms. II. Application example," *IEEE Transaction on Systems, Man and Cybernetics, Part A: Systems and Humans*, 28, pp. 38–47.

[32] Kursawe, F., 1991, "A variant of evolution strategies for vector optimization," *Parallel Problem Solving from Nature, 1st Workshop, PPSN I*, Vol. 496 of Lecture Notes in Computer Science.

[33] Zitzler, E., Deb, K. and Thiele, L., 2000, "Comparison of multiobjective evolutionary algorithms: Empirical results," *Evolutionary Computation*, 8(2), pp. 173–195.

[34] Deb, K., Thiele, L., Laumanns, M., Zitzler, E., 2002, "Scalable multi-objective optimization test problems," *Congress on Evolutionary Computation,* 1, pp. 825-830.

[35] Nebro, A. J., Luna, F., Alba, E., Dorronsoro, B., Durillo, J. J. and Beham, A., 2008, "AbYSS: adapting scatter search to multiobjective optimization," *IEEE Transactions on Evolutionary Computation*, 12, pp. 439-457.

[36] Durillo, J. J., Nebro, A. J., Luna, F., Alba, E., 2008, "Solving three-objective optimization problems using a new hybrid cellular genetic algorithm," *Proceedings of the 10th International Conference on Parallel Problem Solving from Nature*, September 13-17, Technischie Universitat Dortmund, Germany, pp. 661-670.

[37] Eskandari, H. and Geiger, C. D., 2008, "A fast pareto genetic algorithm approach for solving expensive multiobjective optimization problems," *Journal of Heuristics*, 14, pp. 203-241.

[38] Reyes, M., Coello, C.A., 2005, "Improving pso-based multi-objective optimization using crowding, mutation and epsilon-dominance," *Proceedings of Third International Conference on Evolutionary Multi-Criterion Optimization*, Guanajuato, Mexico, March 9-11, pp. 505-519.

[39] Zitzler, E., Laumanns, M. and Thiele, L., 2002, "SPEA2: improving the strength pareto evolutionary algorithm for multiobjective optimization," *Proceedings of the Conference on Evolutionary Methods for Design Optimization and Control*, CIMNE Barcelona, Spain, pp. 95-100.

[40] Deb, K., Sundar, J., Rao, U. B. and Chaudhuri, S., 2006, "Reference point based multi-objective optimization using evolutionary algorithms," *International Journal of Computational Intelligence Research*, 2(3): 273-286.

## Appendix: Tabulated Test Results for 30 runs of each algorithm on each test problem

**Table 1. SCH test problem for 50 function evaluations, for each major cell the first column indicates the average of thirty runs, the second column indicates the standard deviation and the third column indicates the rank**

| SCH 50 | Spread | | | Generalized Spread | | | Generational Distance | | | Inverted Generational Distance | | | Hyper Volume | | | Percentage of Pareto Points | | | Sum of ranks | Over all Rank |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **PSP** | 0.618 | 0.056 | 1 | 0.647 | 0.055 | 1 | 0.0006 | 0.0005 | 1 | 0.0018 | 0.0005 | 1 | 0.8227 | 0.0009 | 1 | 91.1 | 2.47 | 1 | 6 | 1 |
| **CellDE** | 1.000 | 0.000 | 6 | 1.000 | 0.000 | 6 | 974.09 | 1591.8 | 6 | 68.729 | 112.28 | 6 | 0.0000 | 0.0000 | 6 | 2.00 | 0.00 | 6 | 36 | 6 |
| **FastPGA** | 0.965 | 0.102 | 4 | 0.941 | 0.172 | 4 | 116.86 | 430.18 | 5 | 8.2785 | 30.338 | 5 | 0.0989 | 0.1723 | 2 | 2.27 | 0.69 | 4 | 24 | 4 |
| **NSGAII** | 0.931 | 0.142 | 3 | 0.890 | 0.218 | 3 | 20.123 | 22.742 | 3 | 1.5323 | 1.7348 | 3 | 0.0525 | 0.1606 | 5 | 2.47 | 0.86 | 3 | 20 | 3 |
| **OMOPSO** | 0.903 | 0.176 | 2 | 0.860 | 0.247 | 2 | 12.251 | 19.887 | 2 | 0.9172 | 1.4073 | 2 | 0.0872 | 0.2026 | 3 | 2.80 | 1.63 | 2 | 13 | 2 |
| **SPEA2** | 0.993 | 0.036 | 5 | 0.988 | 0.064 | 5 | 46.523 | 80.173 | 4 | 3.3259 | 5.6456 | 4 | 0.0683 | 0.1788 | 4 | 2.07 | 0.37 | 5 | 27 | 5 |

**Table 2. FON test problem for 100 function evaluations, for each major cell the first column indicates the average of thirty runs, the second column indicates the standard deviation and the third column indicates the rank**

| FON 100 | Spread | | | Generalized Spread | | | Generational Distance | | | Inverted Generational Distance | | | Hyper Volume | | | Percentage of Pareto Points | | | Sum of ranks | Final Rank |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **PSP** | 0.718 | 0.141 | 2 | 0.729 | 0.175 | 3 | 0.0737 | 0.0492 | 3 | 0.0149 | 0.0059 | 4 | 0.1098 | 0.0470 | 4 | 6.02 | 2.83 | 5 | 17 | 3 |
| **CellDE** | 0.765 | 0.145 | 6 | 0.741 | 0.191 | 4 | 0.1547 | 0.0702 | 7 | 0.0216 | 0.0073 | 7 | 0.0336 | 0.0383 | 7 | 3.67 | 1.69 | 7 | 31 | 6 |
| **Abyss** | 0.868 | 0.128 | 7 | 0.864 | 0.173 | 7 | 0.1178 | 0.0751 | 6 | 0.0210 | 0.0088 | 6 | 0.0634 | 0.0680 | 6 | 5.28 | 2.46 | 6 | 32 | 7 |
| **FastPGA** | 0.754 | 0.173 | 5 | 0.715 | 0.195 | 2 | 0.0772 | 0.0577 | 4 | 0.0149 | 0.0080 | 3 | 0.1004 | 0.0616 | 5 | 7.40 | 2.79 | 4 | 18 | 4 |
| **NSGAII** | 0.751 | 0.160 | 4 | 0.746 | 0.186 | 5 | 0.0457 | 0.0286 | 2 | 0.0115 | 0.0047 | 2 | 0.1360 | 0.0641 | 2 | 9.07 | 1.76 | 2 | 15 | 2 |
| **OMOPSO** | 0.550 | 0.155 | 1 | 0.574 | 0.159 | 1 | 0.0272 | 0.0151 | 1 | 0.0076 | 0.0045 | 1 | 0.1803 | 0.0468 | 1 | 9.40 | 1.22 | 1 | 5 | 1 |
| **SPEA2** | 0.748 | 0.192 | 3 | 0.770 | 0.210 | 6 | 0.0812 | 0.0741 | 5 | 0.0152 | 0.0080 | 5 | 0.1105 | 0.0799 | 3 | 7.67 | 2.75 | 3 | 22 | 5 |

**Table 3. KUR test problem for 100 function evaluations, for each major cell the first column indicates the average of thirty runs, the second column indicates the standard deviation and the third column indicates the rank**

| KUR 100 | Spread | | | Generalized Spread | | | Generational Distance | | | Inverted Generational Distance | | | Hyper Volume | | | Percentage of Pareto Points | | | Sum of ranks | Final Rank |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **PSP** | 0.654 | 0.130 | 1 | 0.669 | 0.143 | 2 | 0.0216 | 0.0151 | 1 | 0.0033 | 0.0013 | 1 | 0.6530 | 0.0813 | 1 | 8.02 | 2.74 | 1 | 7 | 1 |
| **CellDE** | 0.792 | 0.144 | 6 | 0.829 | 0.187 | 6 | 0.0981 | 0.0607 | 6 | 0.0091 | 0.0030 | 6 | 0.3929 | 0.1155 | 6 | 5.63 | 1.96 | 6 | 36 | 7 |
| **Abyss** | 0.710 | 0.123 | 2 | 0.665 | 0.169 | 1 | 0.1309 | 0.0586 | 7 | 0.0093 | 0.0024 | 7 | 0.3881 | 0.1101 | 7 | 4.07 | 1.20 | 7 | 31 | 5 |
| **FastPGA** | 0.779 | 0.141 | 5 | 0.789 | 0.174 | 5 | 0.0553 | 0.0309 | 4 | 0.0064 | 0.0027 | 4 | 0.5087 | 0.1351 | 3 | 7.27 | 1.80 | 5 | 26 | 4 |
| **NSGAII** | 0.741 | 0.121 | 4 | 0.762 | 0.151 | 3 | 0.0459 | 0.0272 | 2 | 0.0059 | 0.0022 | 2 | 0.5231 | 0.1193 | 2 | 7.50 | 2.06 | 2 | 15 | 2 |
| **OMOPSO** | 0.735 | 0.103 | 3 | 0.788 | 0.132 | 4 | 0.0480 | 0.0362 | 3 | 0.0062 | 0.0024 | 3 | 0.5015 | 0.1173 | 5 | 7.40 | 2.06 | 4 | 22 | 3 |
| **SPEA2** | 0.796 | 0.115 | 7 | 0.835 | 0.135 | 7 | 0.0647 | 0.0578 | 5 | 0.0065 | 0.0024 | 5 | 0.5024 | 0.1230 | 4 | 7.47 | 2.19 | 3 | 31 | 5 |

**Table 4. ZDT6 test problem for 1000 function evaluations, for each major cell the first column indicates the average of thirty runs, the second column indicates the standard deviation and the third column indicates the rank**

| ZDT6 1000 | Spread | | | Generalized Spread | | | Generational Distance | | | Inverted Generational Distance | | | Hyper Volume | | | Percentage of Pareto Points | | | Sum of ranks | Final Rank |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **PSP** | 0.948 | 0.036 | 3 | 0.949 | 0.044 | 4 | 2.5791 | 0.4629 | 5 | 0.2172 | 0.0067 | 7 | 0.0000 | 0.0000 | 2 | 0.81 | 0.27 | 2 | 23 | 4 |
| **CellDE** | 0.927 | 0.033 | 1 | 0.921 | 0.051 | 1 | 2.7466 | 0.7098 | 7 | 0.1935 | 0.0195 | 4 | 0.0000 | 0.0000 | 2 | 0.63 | 0.24 | 6 | 21 | 3 |
| **Abyss** | 0.965 | 0.043 | 6 | 0.942 | 0.080 | 3 | 2.7421 | 0.9406 | 6 | 0.1540 | 0.0203 | 2 | 0.0000 | 0.0000 | 2 | 0.31 | 0.24 | 7 | 26 | 6 |
| **FastPGA** | 0.974 | 0.033 | 7 | 0.970 | 0.039 | 7 | 2.2004 | 0.5807 | 2 | 0.1821 | 0.0186 | 3 | 0.0000 | 0.0000 | 2 | 0.66 | 0.25 | 5 | 26 | 7 |
| **NSGAII** | 0.949 | 0.027 | 4 | 0.951 | 0.038 | 5 | 2.2715 | 0.4127 | 3 | 0.1941 | 0.0093 | 5 | 0.0000 | 0.0000 | 2 | 0.76 | 0.24 | 4 | 23 | 5 |
| **OMOPSO** | 0.962 | 0.130 | 5 | 0.958 | 0.229 | 6 | 1.2942 | 0.4523 | 1 | 0.0311 | 0.0489 | 1 | 0.0725 | 0.1207 | 1 | 1.29 | 0.51 | 1 | 15 | 1 |
| **SPEA2** | 0.942 | 0.028 | 2 | 0.940 | 0.054 | 2 | 2.3790 | 0.5212 | 4 | 0.2018 | 0.0090 | 6 | 0.0000 | 0.0000 | 2 | 0.80 | 0.28 | 3 | 19 | 2 |

**Table 5. DTLZ1 test problem for 500 function evaluations, for each major cell the first column indicates the average of thirty runs, the second column indicates the standard deviation and the third column indicates the rank**

| DTLZ1 500 | Spread | | | Generalized Spread | | | Generational Distance | | | Inverted Generational Distance | | | Hyper Volume | | | Percentage of Pareto Points | | | Sum of ranks | Final Rank |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **PSP** | 0.687 | 0.067 | 3 | 0.647 | 0.077 | 2 | 50.786 | 6.3621 | 5 | 0.5482 | 0.1233 | 3 | 0.0000 | 0.0000 | - | 7.64 | 1.65 | 1 | 14 | 1 |
| **CellDE** | 0.662 | 0.111 | 1 | 0.593 | 0.134 | 1 | 42.929 | 10.831 | 4 | 0.6194 | 0.1170 | 4 | 0.0000 | 0.0000 | - | 3.67 | 0.98 | 6 | 16 | 2 |
| **Abyss** | 0.862 | 0.130 | 7 | 0.883 | 0.184 | 7 | 30.240 | 12.584 | 2 | 0.4281 | 0.1875 | 1 | 0.0000 | 0.0000 | - | 4.48 | 1.34 | 5 | 22 | 5 |
| **FastPGA** | 0.809 | 0.088 | 6 | 0.757 | 0.123 | 6 | 42.668 | 9.4050 | 3 | 0.4735 | 0.1438 | 2 | 0.0000 | 0.0000 | - | 5.87 | 1.54 | 4 | 21 | 4 |
| **NSGAII** | 0.730 | 0.102 | 4 | 0.685 | 0.112 | 4 | 51.582 | 7.6057 | 6 | 0.6666 | 0.2400 | 6 | 0.0000 | 0.0000 | - | 7.11 | 1.70 | 2 | 22 | 6 |
| **OMOPSO** | 0.664 | 0.096 | 2 | 0.670 | 0.086 | 3 | 28.876 | 3.7628 | 1 | 0.6657 | 0.1529 | 5 | 0.0000 | 0.0000 | - | 2.84 | 0.70 | 7 | 18 | 3 |
| **SPEA2** | 0.730 | 0.073 | 5 | 0.689 | 0.097 | 5 | 58.040 | 9.9989 | 7 | 0.7074 | 0.2274 | 7 | 0.0000 | 0.0000 | - | 6.19 | 1.59 | 3 | 27 | 7 |