A Parallel Evolutionary System for Multi-objective Optimisation

1st Mohammad Hamdan
MACS
Heriot Watt University
Dubai, UAE
m.hamdan@hw.ac.uk

2nd Gunter Rudolph
Chair of Algorithm Engineering
Technische Universitat Dortmund
Dortmund, Germany
Guenter.Rudolph@tu-dortmund.de

3rd Nicola Hochstrate
Chair of Algorithm Engineering
Technische Universitat Dortmund
Germany, Dortmund
nicola.hochstrate@tu-dortmund.de

***Abstract*—Parallel evolutionary algorithms have been used for solving multiobjective optimization problems. The aim is to find or approximate the Pareto optimal set in a reasonable time. In this work, we present a new approach that divides the objective search-space into different partitions and assigns each processor its corresponding partition. Each processor will try to find the set of solutions for its partition only. The sub-Pareto fronts will be combined later and the parallelisation approach is based on a mutli-start approach by having independent algorithm on every processor with its own starting points. Experimental results on well known test cases showed that the proposed method outperformed several state-of-the-art evolutionary algorithms regarding convergence to the true Pareto front and gave very competitive results when considering the hypervolume metric. Also, superlinear speedup results were achieved for all test functions.**

***Index Terms*—Multiobjective Optimisation, Evolutionary Multiobjective Algorithms, Parallelisation, Objective Search Space**

I. Introduction

In single objective optimization we are interested in a single solution that minimizes or maximizes a given problem. However, many real world problems have more than one objective function to be considered during the optimization process (known as multi-objective or criteria optimization). For multi-objective problems (MOPs) there is a set of solutions rather than one unique optimal solution. The set of optimal solutions is called the Pareto optimal set. The solutions are all possible tradeoffs for the MOP taking into account all objectives. When plotted in the objective space they give the Pareto-front. Also, these solutions are incomparable to one another. In other words, no solution dominates another solution in the Pareto set. Thus they form a set of non-dominated solutions. Formally [12], a solution $x^* = x_1, x_2, ..., x_n$ dominates a solution $y^* = \{y_1, y_2, ..., y_n\}$ (denoted as x $\prec$ y) if and only if $\forall i \in \{1, ..., n\}, x_i \leq y_i$ and $\exists i \in \{1, ..., n\}, x_i < y_i$.

In this work, we look at how to use parallel processing [5] for dividing the objective search space of multi-objective problems and assigning each partition to a different processor. The motivation for dividing the objective space using a geometric-based apporach is to guarantee that each processor is working on a different part of the Pareto front, thus processors can work independently, otherwise all processors will converge to the same single solution. On each processor we execute an evolutionary algorithm using a multi start approach. Here we run a single evolutionary algorithm on every processor but with different parts of the objective space using additional constraints.

Figure 1 illustrates this concept for an artificial bi-objective optimization problem. The first objective function was used for the division of the objective space. Each processor will be given one additional constraint that specifies its partition and several starting points retrieved from the local archive that was built during the optimization process. This explicit division of the objective space infers an implicit division of the search space since all starting points given to a certain processor will be within a certain lower and upper bounds for every decision variable.

We engineered state-of-the-art single and multi-objective evolutionary algorithms in a parallel system called PESMO. PESMO stands for Parallel Evolutionary System for Multi-objective Optimisation. It comprises three new features: 1) an approach for generating and retrieving the starting points for the different partitions given to the set of processors. This was achieved by using a single objective optimizer for every objective function and thus building an archive of solution vectors, 2) a well defined geometric-based approach for dividing the objective space of MOPs, 3) Deciding at runtime the objective function to be used for dividing the objective space into different partitions.

An attempt to divide the objective space was presented in [11]. However, they specify manually a point to be used as a reference for giving boundaries to processors. The use of reference points requires previous knowledge of every single problem in advance before applying the parallelisation. In this paper, we automate this process by determining the boundaries of the Pareto front and dividing the objective space at run time regardless of the number of processors or problem type. Others [1] looked at binary quadratic programming problems or parallel algorithms for Knapsack [2].

The rest of the article is structured as follows. The background and related work is given in Section II. In Section III, we give a thorough description of the proposed approach. The experimental environment is presented in Section IV and finally we conclude and outline future work in Section V.

978-1-7281-6929-3/20/$31.00 ©2020 IEEE

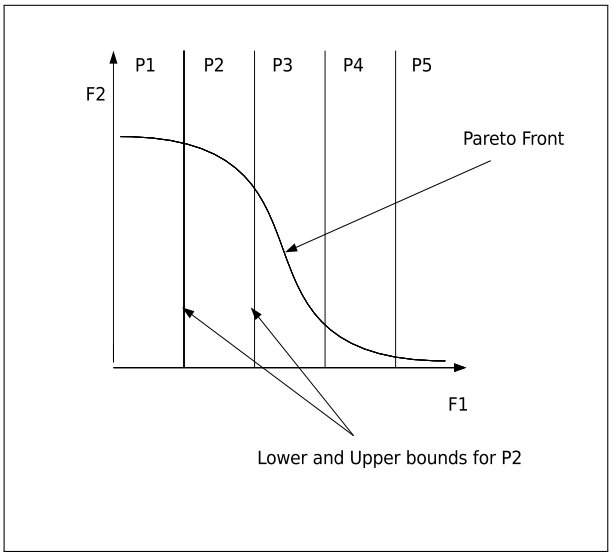

Fig. 1. The division of an objective space of an artificial bi-objective problem on five processors. F1 and F2 are dummy functions.

II. Background and Related Work

This section describes the single and multiple objective evolutionary algorithms [21] [20] used in PESMO.

A. $(1+1)$-CMA-ES

The (1+1)-CMA-ES [22] which is part of the Shark library [24] [1] is a co-evolutionary single objective elitist algorithm. It is based on two basic concepts: 1) *derandomisation* of the mutation distribution as it adapts in a deterministic way, 2) *cummulation* where the search path of the previous population is considered. The (1+1)-CMA-ES has been modified so that it keeps track of the traces found during the search process. These traces will be used later by the parallel system for finding starting points for different regions of the objective space.

B. SMS-EMOA

The S-metric Selection-EMOA (SMS-EMOA) [9] aims at maximizing the S-metric value of the population. This optimization aim rewards progression toward the Pareto front as well as a good distribution of individuals. The maximal S-metric value is reached by the Pareto front. Thus, optimizing the S-metric value is very general purpose. It follows a steady-state selection scheme and an equiprobable mating selection are applied. he runtime of a generation of SMS-EMOA is $O(\mu^{d/2+1})$ as described in [10]. It has a dynamic selection mechanism where it can start with any initial population size and evolve until reaching the desired population size.

C. Overview of Related Parallel Evolutionary Multi-objective Optimization Algorithms

In the following subsections, selected different approaches for parallelising Evolutionary Multi-objective Optimization Algorithms (EMOAs) that either divide the decision/objective search spaces directly or indirectly are outlined.

1) Dividing the Decision Search Space: The work of [14], [23] are similar in principle. The population is divided into subpopulations. Each processor gets a subpopulation. To divide the population it is sorted according to an objective function (the objective functions are used in turn). This causes similar individuals to be grouped together. Therefore, this would divide the search space indirectly and each processor would search different areas of the search space. The problems with these two approaches are that the high possibility of producing infeasible solutions (i.e. individuals that are outside the search space of a given processor) and causing high communication overhead for sending them to other processors.

In [25], an approach called Multiple Resolution Multi-Objective Genetic Algorithm (MRMOGA), the island model is used as the underlying parallel system. A key feature in MRMOGA is the possibility of encoding solutions using different resolutions. Here, each island has its population and uses a different resolution for its encoding. Results are shown only for the first three ZDT test cases. Also, this approach has poor distribution of the results where distances among solutions are not equal.

In [36], the divide-and-conquer technique is used to divide the decision variables space into small components. Here, if there are m variables in the decision variable space then m sub-populations will be created. Each sub-population will focus on optimizing one decision variable. The sub-populations are further divided into groups and each group will be assigned to a node. The sub-populations work in a cooperative co-evolutionary manner. However, this approach might have problems with scalability as the number of generated sub-populations is equal to the number of decision variables.

2) Dividing the Objective Search Space: In [11] an approach called cone separation was introduced to divide the objective space into sub-spaces. Each processor is assigned a sub-space (region). To define the boundaries of a given region, each processor is given constraints according to the borders of its boundaries. However, each processor explores the entire search space and solutions outside the region of a given processor are considered infeasible and migrated to other processors every generation which cause extra communication overhead. In our approach, we use an efficient adaptive approach for determining the shape of the Pareto front which does not require lots of communication overhead.

The cone separation idea was further improved in [35] using clustering algorithms and an island model. Here, the geometric subdivision scheme in cone separation is replaced with a k-means clustering algorithm where k is set to the number of processors. The experiments showed that the objective space clustering is better the than search space clustering. Also, when the zone constraints are deactivated the results improved.

To sum up, the objective-based approaches divide the objective space geometrically or cluster the population. Both approaches add constraints. We think that an objective-based approach would have better scalability with many objectives.

[1] Available for download from http://shark-project.sourceforge.net/

Also, the objective space can be divided based on information by Pareto bounds thus yielding even smaller search space.

III. Description of The Proposed Approach for Dividing the Objective Search Space

The aim of this work is to approximate the Pareto-front by dividing the objective space into P partitions and assigning each partition to a different processor. This explicit division should result in an implicit division of the search space. especially if the problem is regular. By regular there is a coresponding clear division of the search space as a result of dividing the objective space. In other words, each processor would have new lower and upper bounds for every decision variable of the optimization problem.

Another important issue is to give each processor starting points that satisfy the lower and upper bounds for its region boundaries in the objective space. It is very difficult and time consuming to generate a random point and then check if its functions' values satisfy a certain partition. Therefore, we build an archive of starting points that can be used later according to the needs of each processor. The archive contains non-dominated and dominated points. Also, all processors are working using a multi-start approach and use collective communication.

PESMO comprises four phases and an outline is given as follows: 1) For each objective function find the solution vector that minimizes it using the $(1+1)$-CMA-ES. Use these solution vectors to initialize the $(1+1)$-CMA-ES on different processors. Each processor will choose a vector and different weights. The aim is to build an archive of starting points from the traces of the optimization process.

The traces of all the processors are gathered on the root processor. The traces are filtered into dominated and non-dominated points then broadcasted to the other processors. 2) The ranges matrix is built at the root processor and all needed values are calculated from the matrix. 3) Next, count how many non-dominated points are within the min and max values that were found for every objective function (found in each row of the ranges matrix). The objective function with the biggest count will be used to divide the objective space. Its corresponding values are then broadcasted to the rest of the processors. 4) Each processor will search its archive of starting points to initialize SMS-EMOA and use it for the remaining function evaluations. Finally, the root processor will gather and combine all sub-Pareto fronts. The four phases are described next.

1) Phase I: The first phase is described in Algorithm 1. It works as follows: Each processor (up to d processors) will run the $(1+1)$-CMA-ES single objective optimizer described in Section II-A for one of the objective functions in the problem. The $(1+1)$-CMA-ES will then return a solution vector X_i^* that minimizes F_i. Next, use these solution vectors to initialize the $(1+1)$-CMA-ES used on different processors. Each processor will use a different solution vector with different weights. Here, the Tchebycheff method [28] is used to generate the traces on different processors. We adjust the method slightly as we set a very small weight for the objective function that was used to initialize the $(1+1)$-CMA-ES. Also, we vary the vectors and weights on different processors so that there is good variety in starting points.

We have modified $(1+1)$-CMA-ES so that copies of the traces found during the optimization process are stored. Each processor will generate its own traces that are converted into a sub-population. All sub-populations are gathered on the root processor and combined into one big population. The population is filtered into non-dominated and dominated population then broadcasted to all processors to be stored in the local archive shown. The local arcvhive has traces of decision variables and the corresponding function value per optimisation function. About 10% MaxFEs will be used for finding the function minimum and building archive on each processor during this phase. The size of the local archive on each processor in bytes can be calculated using Equation 1.

$$ArchiveSize = (Dom + NonDom) * (n + d) * w \quad (1)$$

Algorithm 1 Pseudo-code for phase I: Find function minimum and build the archive.

Each Processor P_i will do the following:
MaxFEs $\leftarrow \frac{TotalFEs}{P}$
$z_k \leftarrow CMA-ES(4\%MaxFEs, k)$ // k is the objective function
$z_{1\cdots d} \leftarrow$ Gather z on root processor
if root **then** Broadcast vecs to the other processors
end if
Initialize $w_{1\cdots d}$
$u^*_{1\cdots d} \leftarrow z^*_{1\cdots d} - \varepsilon$ // u_i : Utopian point for objective i
$\tau^{(f,w,u^*)}(x) \leftarrow max_{j \leftarrow 1\cdots d} w_j \mid f_j - u^*_j \mid$
$pop_i \leftarrow CMA-ES(6\%MaxFEs, \tau)$ // each processor has its population
if root **then**
    Pop $\leftarrow$ Gather pop_i
    NonDomPop $\leftarrow$ GetPareto(Pop)
    DomPop $\leftarrow$ Pop - NonDomPop
    Broadcast (NonDomPop, DomPop) to other processors
end if
$Archive_1 \leftarrow NonDomPop$
$Archive_2 \leftarrow DomPop$
$ArchiveSize_1 \leftarrow Size(NonDomPop)$
$ArchiveSize_2 \leftarrow Size(DomPop)$

2) Phase II: The aim of the second phase is to build the ranges matrix shown in Figure 2. The number of function evaluations needed here are $d * d$. An outline of the needed steps is shown in Algorithm 2. The values $(X_i^*, i = 1 \cdots d)$ found in the previous phase will be used here. The matrix will be used to give an approximation of the bounds of the Pareto-front in the objective space. For every single row in the ranges matrix we do find the minimum $\tilde{m}_i$ and maximum $\tilde{M}_i$ values. The difference $\tilde{D}_i$ is $\tilde{M}_i - \tilde{m}_i$. $Diff_i$ corresponds to the difference between the lowest and highest values for objective

$$\begin{array}{cccc} f_1(x_1^*) & f_1(x_2^*) & \dots & f_1(x_d^*) \\ f_2(x_1^*) & f_2(x_2^*) & \dots & f_2(x_d^*) \\ & \vdots & & \\ f_d(x_1^*) & f_d(x_2^*) & \dots & f_d(x_d^*) \end{array}$$

Fig. 2. The ranges matrix used to define an approximation of the Pareto bounds in the objective search space.

i in the ranges matrix. When $Diff_i$ is multiplied with constant c we can control the area to work on in the objective search space. The default value for c is zero. The value $\tilde{D}_i$ will be used to divide the objective space (according to objective function i) into P partitions with width not exceeding δ_i. All of these values $m_i, \delta_i, and\, i, \forall i_{\leftarrow i \cdots d}$ are broadcasted to other processors.

Once other processors receive the required values then the question is which objective function the parallel system should use to the divide the objective space (i.e. objective functions $1 \cdots d$). Therefore, the next phase will look at this issue.

Algorithm 2 Pseudo-code for phase II: Build the ranges matrix.

```
Each processor P_i will do the following:
if root then
    for i ← 1 ··· d do
        for j ← 1 ··· d do
            RangesMatrix_{ij ← f_i(X_j^*)}
        end for
        m_i ← Min(RangesMatrix_{i_{j ← 1 ··· d}})
        M_i ← Max(RangesMatrix_{i_{j ← 1 ··· d}})
        Diff_i ← M_i − m_i
        M̃_i ← M_i + c · Diff_i
        m̃_i ← m_i − c · Diff_i
        D̃_i ← M̃_i − m̃_i
        δ_i ← D̃_i / P
    end for
end if
```

3) Phase III: As mentioned earlier, we need to decide which objective function should be used to divide the objective space into P partitions. There are d possibilities. We call this process partition probing and it is described in Algorithm 3. One option is to use the objective function with biggest $\tilde{D}$ value. However, for few irregular problems it is possible that we have more than one solution to the problem (multi-modal or non-unique solutions). In this case the range from $\tilde{m}$ to $\tilde{M}$ could cover only part of the Pareto front and not all of it. If this range is partitioned into P partitions then it is possible that few (or many) processors are doing useless work.

Therefore, each processor will search its local archive for the non-dominated points that are within the processor's constraint for all objective functions in turn. The points will be counted for all objectives. The counter will be collected on the root processor and the overall sum will be calculated. The objective function that has the maximum number of non-dominated points will be used for dividing the objective space.

Algorithm 3 Pseudo-code for phase III: Find the objective function to divide the objective space.

```
Each processor P_i will do the following:
for k ← 1 ··· d do
    Broadcast to all processors m̃_k, δ_k, and k
    A_i ← m̃_k + i · δ_k − ε
    B_i ← m̃_k + (i + 1) · δ_k + ε
    for x ← 1 ··· ArchiveSize_1 do
        if A_i ⩽ Archive_1 → Trace_x → F_k ⩽ B_i then NonDomCnt_ik + +
        end if
    end for
    if root then Gather NonDomCnt_ik
    end if
end for
if root then
    NonDomSum_k ← Σ_{i=0}^{P} NonDomCnt_ik
end if
Find k with max NonDomSum_k
Broadcast m̃_k, δ_k, and k
```

4) Phase IV: The final phase is outlined in Algorithm 4. The root processor broadcasts to all processors the needed information in order to divide the objective space. All processors will search their local archive for starting points that are within the processor's additional constraints. Execute SMS-EMOA with the initial population created from the traces in the archive on each processor then gather the sub-populations on the root processor.

The following strategy (which gives priority to non-dominated points) is used when searching for points in the local archive: 1) Search for non-dominated points in the non-dominated points archive first, 2) Stop the search when number of non-dominated points found reaches MaxSP, 3) If variable $useDom$ is set true or no points are found so far then search in dominated points archive, 4) If still no points are found then each processor will increase its area by 50%.

IV. Experimental Environment

The parallel machine used is a cluster of Linux machines running Red Hat linux. All processors are homogeneous with local memory and cache. The machines are connected through a standard ethernet switch. No special hardware or software is used to minimize communication overhead. The standard MPI library [31] and gcc compiler were used.

The classical family of ZDT1 to ZDT4 and ZDT6 MOPs [40] were used to test the proposed PESMO. Also, the following well known state-of-the-art EMOAs: AbYSS [30], NSGA2 [16], PAES [26], PESA-II [13], SPEA2 [41], MOCELL [29] and MOEA/D [39] were used for comparison purposes. They are all implemented in the JMetal Framework [19]. The generational distance (GD) [37] and hypervolume

Algorithm 4 Pseudo-code for phase IV: Search the archive for starting points and run EMOA on each processor P_i

Each processor P_i will do the following:
receive from root $\tilde{m}_k, \delta_k, k$
$A_i \leftarrow \tilde{m}_k + i \cdot \delta_k - \epsilon$, $B_i \leftarrow \tilde{m}_k + (i+1) \cdot \delta_k + \epsilon$
if rank == 0 **then** $A_i \leftarrow A_i - \delta_k$
end if
if rank == Size - 1 **then** $B_i \leftarrow B_i + \delta_k$
end if
while $SP_{cnt} == 0$ **do**
  for $x \leftarrow 1 \cdots ArchiveSize_1$ **do**
    if $A_i \leqslant Archive_1 \rightarrow Trace_x \rightarrow F_k \leqslant B_i$ **then**
      $StartingPoint_{SP_{cnt}} \leftarrow Archive_1 \rightarrow Trace_x$, SP_{cnt++}
      if $SP_{cnt} \geqslant MaxSP$ **then** Break
      end if
    end if
  end for
  if $SP_{cnt} == 0 \,||\, useDom == true$ **then**
    for $x \leftarrow 1 \cdots ArchiveSize_2$ **do**
      if $A_i \leqslant Archive_2 \rightarrow Trace_x \rightarrow F_k \leqslant B_i$ **then**
        $StartingPoint_{SP_{cnt}} \leftarrow Archive_2 \rightarrow Trace_x$
        if $SP_{cnt++} \geqslant MaxSP$ **then** Break
        end if
      end if
    end for
  end if
  if $SP_{cnt} == 0$ **then**
    $Delta \leftarrow \frac{B_i - A_i}{4}$
    $A_i \leftarrow A_i - Delta$, $B_i \leftarrow B_i + Delta$
    for $x \leftarrow 1 \cdots ArchiveSize_*$ **do**
      if $A_i \leqslant Archive_* \rightarrow Trace_x \rightarrow F_k \leqslant B_i$ **then**
        $StartingPoint_{SP_{cnt}} \leftarrow Archive_* \rightarrow Trace_x$, SP_{cnt++}
        if $SP_{cnt} \geqslant MaxSP$ **then** Break
        end if
      end if
    end for
  end if
end while
$FEs \leftarrow MaxFES - 10\%MaxFES - EA_{FEs} - \frac{d*d}{P}$
LocalPop$_i \leftarrow EMOA(A_i, B_i, k, StartingPoints,$
$SP_{cnt}, \frac{popSize}{P}, n, d, FEs)$
if root **then** Gather LocalPop$_i$
end if

(HV) [42] metrics are used to compare the quality of obtained solutions for all algorithms. Smaller GD and higher HV values are desirable. PESMO and the other EMOAs were setup as follows. The population size was set to 100, crossover operator is simulated binary (SBX) [4], mutation operator is polynomial [3], crossover probability ($P_c = 0.9$), mutation probability ($P_m = 1/n$), crossover distribution index ($\eta_c = 20$), mutation distribution index ($\eta_m = 20$). In all experiments we used 25000 function evaluations with 40 runs. The maximum number of starting points was set to 100 for PESMO. The population size and number of function evaluations are fixed as this is a standard for comparison with other techniques as reported in the literature. Table I present the GD metric results for ZDT MOPs while Table II show the HV metric results that are discussed next.

A. Activating and Deactivating the Constraints

Once the configuration of the various parameters are determind we can perform the required experiments in order to test the proposed methodology implemented in PESMO. The aim of this set of experiments is to see the effect of deactivating the constraints at a certain point of program execution on the quality of obtained solution. Once the constraints are deactivated then each processor can explore the entire objective space and different parts of the Pareto front can be explored by more than one processor (normally the neighboring ones). Using this technique more than one processor can work on the same part of the Pareto front. However, no information exchange is allowed between processors. The question is when to deactivate the constraints. Therefore, we deactivate the constraints after 50%, 75%, 80%, 85%, 90% and 95% FEs. We also experiment with activating the constraints (activate until 100%) and deactivating the constraints for all FEs (De-activate 100%FEs).

Regarding the ZDT1 MOP, PESMO progressively improve the GD metric as the number of processors is increased up to 10. Also, the GD metric improve when the constraints are de-activated in the last 5-25% FEs. When the constraints are deactivated from the beginning of the execution (deactivate 100%) does not seem to improve convergence to Pareto front. This could be due to generating solutions that are dominated by solutions found by other processors. The hypervolume results do not improve as we increase the number of processors. In fact the HV metric slightly decreases as more processors are used. This due to the nature of SMS-EMOA as maximizing the hypervolume across the entire Pareto front by one processor would give slightly bigger dominated area. In the parallel system each processor will use SMS-EMOA to maximize the hypervolume for its sub-Pareto front only. When these points are collected on one processor to generate the final solution. Then, the corresponding HV would decrease as more processors are used despite the fact the GD metric improves. Nonetheless, the HV results obtained by PESMO are still better than other EMOAs as it will be shown in Section IV-B. Also, de-activating the constraints gives a gradual decrease in hypervolume. We think this is due to generating a small overlap between processors and maximizing the HV metric will not by maximized as desired in overlapped regions.

For the ZDT2 MOP, the GD metric slightly improves as number of processors is increased. Also, deactivating the constraints in the last 5-25 % FEs improves the GD metric. However, it slightly gets worse in case of deactivating the

constraints in the last 50% or 100% FEs. This could be due to the fact that SMS-EMOA prefers convex problems and ZDT2 is a non-convex problem [9]. Also, increasing the number of processors improve the results up to 8 processors. Similar behavior to ZDT1 when considering the HV metric.

It is interesting to note that for the ZDT3 MOP, the earlier the constraints are de-activated the better GD results are obtained up to 8 processors. Recall that the ZDT3 Pareto Front shape is disconnected. Therefore, it is better for each processor to explore the entire objective search space, otherwise, few processors might generate a sub-Pareto front that parts of it are dominated by other solutions generated by other processors. This would happen when all sub-Pareto fronts are gathered and combined at the root processor. An interesting behavior is when the constraints are activated 100% FEs as PESMO managed to get very good GD value on exactly 8 processors. This is due to good partitioning of the Pareto front across the 8 processors. We think this behavior is only with this instance of the ZDT3 MOP.

However, the hypervolume results are acceptable only if the constraints are deactivated only in the last 5-25% FEs. This suggests that for disconnected problems it is good to de-activate the constraints only in the last remaining FEs, otherwise, the generated solutions will dominate each other thus reducing the quality of the HV metric values.

The ZDT4 is a challenging MOP as it has 21^9 local Pareto fronts. Thus it is likely for an EMOA to get trapped in local optima. PESMO results for the GD and HV metrics get worse as more processors are used. It is worth noting that deactivating the constraints 100% gave slightly better results than activating the constraints 50-100% FEs. This poor performace of PESMO suggest that separating local Pareto fronts with constraints in the objective space is a difficult task. Therefore, in Section IV-C a new option will be used with PESMO to tackle multi-modal problems.

The results for the ZDT6 MOP confirm the conclusion that de-activating the constraints improve the GD metric but not the hypervolume metric. Also, increasing the number of processors improve the GD metric only of the constraints are de-activated in the last 5-20% FEs up to 8 processors. Recall that ZDT6 MOP has a non-uniform spread of solutions across the Pareto front. Thus, de-activating the constraints helps in giving better approximation of the Pareto front.

A conclusion can be drawn from the above experiments as follows. When solving a new MOP, the shape of the Pareto-front is not known. Therefore, deactivating the constraints at some point (but not too early) would guarantee the robustness of PESMO for many MOPs. Since today's Desktop machines can have 2, 4 or 8 cores then PESMO can be used widely for such type of machines and standard parallel clusters available widely in many institutions.

B. Comparing PESMO with Other EMOAs

The aim is to compare PESMO with few EMOAs. PESMO used with constraints activated only 90% FEs. Also, to provide

TABLE I
GENERATIONAL DISTANCE RESULTS FOR ZDT1, ZDT2, ZDT3 AND ZDT6 MOPS USING SEVERAL EMOAS. PESMO ACTIVATE UNTIL 90% FES. THE SUPERSCRIPTS A, B AND C INDICATE WHETHER PESMO USING MORE THAN TWO PROCESSORS IS STATISTICALLY SIGNIFICANTLY BETTER THAN THE SMS-EMOA AND PESMO_1 AND MOEAD, RESPECTIVELY.

EMOA	GD $\tilde{x}_{IQR}$	EMOA	GD $\tilde{x}_{IQR}$
ZDT1			
AbYSS	$1.82212e-04_{2.99e-05}$	$\text{PESMO}_2^{a,b}$	$6.07875e-05_{8.93e-06}$
NSGA2	$2.21422e-04_{4.86e-05}$	$\text{PESMO}_3^{a,b}$	$5.60789e-05_{1.06e-05}$
PAES	$2.31559e-04_{1.33e-04}$	$\text{PESMO}_4^{a,b}$	$5.62754e-05_{9.87e-06}$
SPEA2	$2.21736e-04_{3.23e-05}$	$\text{PESMO}_5^{a,b}$	$5.49454e-05_{1.62e-05}$
MOEAD	$1.25554e-03_{7.81e-04}$	$\text{PESMO}_6^{a,b}$	$5.97481e-05_{1.41e-05}$
MOCELL	$1.76603e-04_{1.85e-05}$	$\text{PESMO}_7^{a,b}$	$5.35609e-05_{1.08e-05}$
PESAII	$2.21983e-04_{8.21e-05}$	$\text{PESMO}_8^{a,b}$	$5.48323e-05_{1.08e-05}$
SMS-EMOA^b	$7.00101e-05_{1.90e-05}$	$\text{PESMO}_9^{a,b}$	$5.31404e-05_{1.40e-05}$
PESMO_1	$7.97139e-05_{1.39e-05}$	$\text{PESMO}_{10}^{a,b}$	$5.07411e-05_{1.19e-05}$
ZDT2			
AbYSS	$1.04149e-04_{5.83e-05}$	$\text{PESMO}_2^{a,b}$	$4.39822e-05_{3.29e-06}$
NSGA2	$1.68839e-04_{4.30e-05}$	$\text{PESMO}_3^{a,b}$	$4.32627e-05_{5.63e-06}$
PAES	$2.34300e-04_{3.46e-04}$	$\text{PESMO}_4^{a,b}$	$4.29589e-05_{5.18e-06}$
SPEA2	$1.67339e-04_{6.17e-05}$	$\text{PESMO}_5^{a,b}$	$4.35050e-05_{3.70e-06}$
MOEAD	$1.35862e-03_{8.51e-04}$	$\text{PESMO}_6^{a,b}$	$4.26783e-05_{5.89e-06}$
MOCELL	$6.11599e-05_{3.21e-05}$	$\text{PESMO}_7^{a,b}$	$4.37754e-05_{3.63e-06}$
PESAII	$1.76887e-04_{1.00e-04}$	PESMO_8	$4.51888e-05_{5.65e-06}$
SMS-EMOA	$4.56165e-05_{2.90e-06}$	PESMO_9	$4.49604e-05_{6.69e-06}$
PESMO_1	$4.63651e-05_{6.06e-06}$	PESMO_{10}	$4.62933e-05_{7.40e-06}$
ZDT3			
AbYSS	$1.93836e-04_{2.24e-05}$	PESMO_2	$1.60390e-04_{1.44e-05}$
NSGA2	$2.12399e-04_{1.70e-05}$	PESMO_3	$1.60482e-04_{2.24e-05}$
PAES	$1.99370e-04_{1.27e-04}$	PESMO_4	$1.62652e-04_{1.08e-05}$
SPEA2	$2.32443e-04_{1.88e-05}$	PESMO_5	$1.81478e-04_{3.27e-05}$
MOEAD	$2.87742e-03_{1.23e-03}$	PESMO_6	$1.75932e-04_{1.97e-05}$
MOCELL	$2.01641e-04_{1.85e-05}$	PESMO_7	$1.67162e-04_{2.12e-05}$
PESAII	$1.92501e-04_{3.84e-05}$	PESMO_8	$1.59114e-04_{2.58e-05}$
SMS-EMOA	$1.46639e-04_{6.43e-06}$	PESMO_9	$1.64333e-04_{1.32e-05}$
PESMO_1	$1.48016e-04_{1.02e-05}$	PESMO_{10}	$1.61850e-04_{2.42e-05}$
ZDT6			
AbYSS	$5.49989e-04_{2.13e-05}$	$\text{PESMO}_2^{a,b,c}$	$5.22343e-04_{6.49e-05}$
NSGA2	$9.94321e-04_{1.28e-04}$	PESMO_3^{b}	$5.33571e-04_{7.76e-05}$
PAES	$8.60327e-04_{8.54e-03}$	$\text{PESMO}_4^{a,b,c}$	$5.17071e-04_{8.48e-05}$
SPEA2	$1.70983e-03_{3.34e-04}$	PESMO_5	$5.27560e-04_{7.77e-05}$
$\text{MOEAD}^{a,b}$	$5.33020e-04_{5.37e-06}$	PESMO_6	$5.57301e-04_{9.33e-05}$
MOCELL	$6.54748e-04_{4.00e-05}$	PESMO_7^{b}	$5.37795e-04_{8.12e-05}$
PESAII	$8.77987e-04_{5.52e-03}$	PESMO_8	$5.70250e-04_{1.03e-04}$
SMS-EMOA^b	$5.43154e-04_{1.50e-05}$	PESMO_9	$5.46228e-04_{4.97e-05}$
PESMO_1	$5.50230e-04_{2.20e-05}$	PESMO_{10}	$5.71132e-04_{7.38e-05}$

results with confidence a non-parametric statistical test (two-sided wilcoxon rank sum test [33]) is performed to find if PESMO is statistically better (with $p < 0.05$) that the best result of the sequential EMOAs (only compare to the best including SMS-EMOA).

PESMO outperformed all EMOAs for the ZDT1 MOP up to 10 processors regarding the GD metric. Also, up to 4 processors it outperformed other EMOAs and was very competitive with SMS-EMOA regarding the HV metric. For the ZDT2 MOP, PESMO gave excellent behavior regarding GD metric up to 7 processors. Beyond this size it was very competitive. The HV metric was also acceptable. Both GD and HV metrics for ZDT3 are very competitive with SMS-EMOA. SMS-EMOA ranked after MOEAD for ZDT6 MOP but PESMO up to 4 processors managed to outperform both regarding the GD metric.

It is worth noting that PESMO on one processor gave better results than SMS-EMOA regarding the HV metric. This suggests that the use of intelligent starting points can improve the results of SMS-EMOA. Thus, the approach proposed

in PESMO for generating starting points is usefull indeed. However, regarding the GD metric no difference was found between SMS-EMOA and PESMO_1.

TABLE II
HYPERVOLUME RESULTS FOR ZDT1, ZDT2, ZDT3 AND ZDT6. PESMO ACTIVATE UNTIL 90% FES.

EMOA	Hypervolume $\tilde{x}_{IQR}$	EMOA	Hypervolume $\tilde{x}_{IQR}$
ZDT1			
AbYSS	$0.661386_{2.30e-04}$	PESMO_2	$0.661719_{2.31e-04}$
NSGA2	$0.659377_{4.82e-04}$	PESMO_3	$0.661489_{2.46e-04}$
PAES	$0.657406_{1.73e-03}$	PESMO_4	$0.661247_{3.00e-04}$
SPEA2	$0.659923_{4.75e-04}$	PESMO_5	$0.661128_{2.99e-04}$
MOEAD	$0.644597_{1.03e-02}$	PESMO_6	$0.660904_{4.67e-04}$
MOCELL	$0.661007_{2.78e-04}$	PESMO_7	$0.660771_{4.46e-04}$
PESAII	$0.657264_{1.03e-03}$	PESMO_8	$0.660707_{2.96e-04}$
SMS-EMOA	$0.662050_{2.41e-05}$	PESMO_9	$0.660528_{3.96e-04}$
PESMO_1^a	$0.662124_{1.86e-05}$	PESMO_{10}	$0.660274_{5.40e-04}$
ZDT2			
AbYSS	$0.328181_{3.68e-04}$	PESMO_2	$0.328151_{8.54e-04}$
NSGA2	$0.326082_{3.89e-04}$	PESMO_3	$0.327832_{8.17e-04}$
PAES	$0.323846_{2.14e-03}$	PESMO_4	$0.327486_{3.31e-03}$
SPEA2	$0.326426_{9.02e-04}$	PESMO_5	$0.327438_{7.84e-04}$
MOEAD	$0.310407_{1.26e-02}$	PESMO_6	$0.327131_{9.66e-04}$
MOCELL	$0.328277_{6.49e-04}$	PESMO_7	$0.327202_{1.68e-03}$
PESAII	$0.324574_{9.40e-04}$	PESMO_8	$0.326342_{4.55e-03}$
SMS-EMOA	$0.328841_{2.57e-05}$	PESMO_9	$0.326266_{3.36e-02}$
PESMO_1^a	$0.328865_{7.88e-03}$	PESMO_{10}	$0.325831_{3.25e-02}$
ZDT3			
AbYSS	$0.515823_{3.46e-03}$	PESMO_2	$0.515772_{1.02e-04}$
NSGA2	$0.514788_{2.02e-04}$	PESMO_3	$0.515485_{2.57e-04}$
PAES	$0.499916_{4.94e-02}$	PESMO_4	$0.515438_{1.60e-04}$
SPEA2	$0.514080_{4.60e-04}$	PESMO_5	$0.515368_{1.77e-04}$
MOEAD	$0.451838_{2.90e-02}$	PESMO_6	$0.514898_{1.39e-04}$
MOCELL	$0.515166_{4.06e-04}$	PESMO_7	$0.514493_{3.40e-04}$
PESAII	$0.512489_{3.73e-03}$	PESMO_8	$0.514319_{3.63e-04}$
SMS-EMOA	$0.516005_{5.84e-05}$	PESMO_9	$0.513734_{3.28e-04}$
PESMO_1^a	$0.516111_{4.12e-06}$	PESMO_{10}	$0.513516_{5.92e-04}$
ZDT6			
AbYSS	$0.400350_{2.32e-04}$	PESMO_2	$0.399319_{5.07e-04}$
NSGA2	$0.388797_{1.89e-03}$	PESMO_3	$0.398209_{3.65e-04}$
PAES	$0.396536_{2.17e-03}$	PESMO_4	$0.397371_{5.01e-04}$
SPEA2	$0.379071_{4.57e-03}$	PESMO_5	$0.396505_{7.74e-04}$
$\text{MOEAD}^{a,b}$	$0.401369_{1.85e-05}$	PESMO_6	$0.395965_{7.27e-04}$
MOCELL	$0.396421_{9.24e-04}$	PESMO_7	$0.395457_{1.04e-03}$
PESAII	$0.396169_{1.01e-03}$	PESMO_8	$0.395459_{1.01e-03}$
SMS-EMOA	$0.400671_{2.21e-04}$	PESMO_9	$0.394663_{8.93e-04}$
PESMO_1^a	$0.400871_{3.48e-04}$	PESMO_{10}	$0.394348_{1.30e-03}$

C. PESMO and Separating Local Pareto Fronts

The poor performance of PESMO with ZDT4 requires adding an option to the parallel algorithm to separate local Pareto fronts efficiently. Therefore, the user can switch on an option that can help PESMO for multimodal problems. It works as follow, deactivate the zone constraints on each processor then every n function evaluations the sub-population of all processors are gathered on the root processor and filtered to non-dominated and dominated solutions. These solutions are broadcasted to each processor where it will select new starting points to be used as initial population according to its zone constraints. The processor then resumes execution until the next gather process is issued.

The user specifies the how many times to gather the sub-populations (assume it is G), then $n = MaxFEs/G$. This has been tested with the following gathering intervals: 2, 4, 8 and 16. In the last interval the zone constraints are activated. The idea behind this technique that if one processor manages to separate the local Pareto fronts and reaches the global Pareto front then other processors should benefit from this. It is possible that a processor will have a shorter path in the objective search space to the PF_{true}. No noticeable extra cost

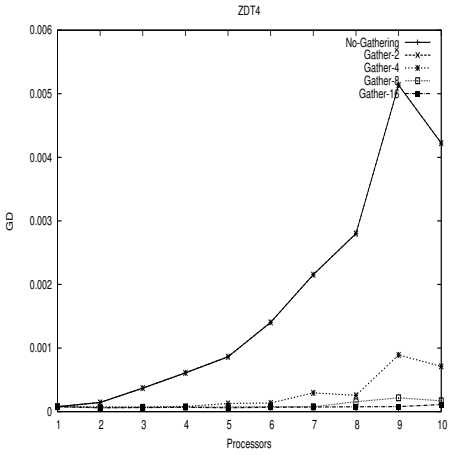

Fig. 3. Median of GD metric for ZDT4 with different gathering intervals.

in terms of execution time due to additional gathering of sub-solutions.

Figure 3 shows the median of the generational distance metric for ZDT4 using different intervals. Due to the complexity of ZDT4 50000 FEs are used. The results show that it is better to use a gathering interval of 8 or more. This confirms that allowing all processors to work on the entire objective search space of a multi-modal problem is useful.

TABLE III
GD RESULTS FOR ZDT4 MOP USING SEVERAL EMOAS WITH 50000FES.

EMOA	GD $\tilde{x}_{IQR}$	EMOA	GD $\tilde{x}_{IQR}$
AbYSS	$2.281\text{e-}04_{9.62e-05}$	$\text{PESMO}_2^{a,b}$	$6.057\text{e-}05_{1.87e-05}$
NSGA2	$1.641\text{e-}04_{5.61e-05}$	$\text{PESMO}_3^{a,b}$	$6.312\text{e-}05_{1.71e-05}$
PAES	$3.033\text{e-}02_{1.96e-01}$	$\text{PESMO}_4^{a,b}$	$6.768\text{e-}05_{1.89e-05}$
SPEA2	$1.782\text{e-}04_{2.94e-05}$	$\text{PESMO}_5^{a,b}$	$6.032\text{e-}05_{1.95e-05}$
MOEAD	$4.909\text{e-}04_{6.46e-04}$	$\text{PESMO}_6^{a,b}$	$6.745\text{e-}05_{2.06e-05}$
MOCELL	$1.633\text{e-}04_{2.61e-05}$	$\text{PESMO}_7^{a,b}$	$7.208\text{e-}05_{4.09e-05}$
PESAII	$2.234\text{e-}04_{6.13e-03}$	PESMO_8^a	$7.872\text{e-}05_{5.35e-05}$
SMS-EMOA	$7.947\text{e-}05_{1.69e-05}$	PESMO_9	$7.968\text{e-}05_{6.33e-05}$
PESMO_1^a	$7.733\text{e-}05_{1.45e-05}$	PESMO_{10}	$1.106\text{e-}04_{8.77e-05}$

Tables III and IV present the generational distance and hypervolume metrics for PESMO and other EMOAs using 50000 FEs. The results show how PESMO can outperform all EMOAs up to 7 processors including SMS-EMOA regarding convergence and very competitive with the hypervolume metric. Also the results up to 10 processors are very competetive.

TABLE IV
HV RESULTS FOR ZDT4 MOP USING SEVERAL EMOAS WITH 50000FES.

EMOA	HV $\tilde{x}_{IQR}$	EMOA	HV $\tilde{x}_{IQR}$
AbYSS	$0.6600_{2.12e-03}$	PESMO_2^b	$0.6617_{2.15e-04}$
NSGA2	$0.6596_{1.29e-03}$	PESMO_3	$0.6615_{4.62e-04}$
PAES	$0.6536_{4.11e-03}$	PESMO_4	$0.6612_{5.49e-04}$
SPEA2	$0.6607_{1.14e-03}$	PESMO_5	$0.6611_{3.63e-04}$
MOEAD	$0.6549_{9.59e-03}$	PESMO_6	$0.6608_{5.19e-04}$
MOCELL	$0.6615_{7.05e-04}$	PESMO_7	$0.6605_{7.37e-04}$
PESAII	$0.6570_{1.88e-03}$	PESMO_8	$0.6602_{1.17e-03}$
SMS-EMOA	$0.6616_{4.69e-04}$	PESMO_9	$0.6599_{1.01e-03}$
PESMO_1^a	$0.6618_{3.77e-04}$	PESMO_{10}	$0.6594_{1.63e-03}$

D. Speedup Analysis

In on order to understand the effect of parallel computing on the behavior of PESMO we need to look at speedup analysis. Recall that SMS-EMOA complexity is $O(\mu^{d/2+1})$ as mentioned in Section II-B. In PESMO the population size μ is divided equally among all processors. Hence, the complexity for PESMO is $O((\frac{\mu}{P})^{d/2+1})$. As each we increase number of processors then SMS-EMOA execution time of every single processor will improve due to reduction in population size. A cost model for PESMO is:

$\mathrm{T}_{PESMO}(P) = \frac{MaxFEs}{P} * (T_{Selection} + T_{Crossover} + T_{Mutate} + T_{Evaluate})$

where $T_{Selection}$ is time to select new population and it depends on population size. In PESMO population size is $\frac{\mu}{P}$, $T_{Evalute}$ is time for one function evaluation, $T_{Crossover}$ and T_{Mutate} are crossover and mutation time respectively. Therefore, increasing number of processors reduces the computation time from two perspectives: 1) dividing the total number of function evaluations across all processors and 2) reducing population size which has a strong effect on selection time. The relative computational speedup is

$$Speedup_P = \frac{Time(PESMO_1)}{Time(PESMO_P)} \quad (2)$$

Therefore, two types of speedup are reported: 1) relative speedup by measuring the time of PESMO after 25000 FEs. This should give the overall behavior of the parallel system and 2) fixed population size speedup where PESMO execution time on one processor is measured with the following population sizes: 100, 50, 34, 25, 20, 17, 14, 11 and 10 to be used with PESMO execution time measured on 1, 2, 3, 4, 5, 6, 7, 8 ,9 and 10 processors, respectively. Hopefully, this will separate the algorithmic behavior of SMS-EMOA due to the high selection time.

Figure 4 presents the average relative speedup and fixed population size speedup for all problems. The relative speedup results are superlinear. This is due to the reduction of population size as the number of processors increase and the calculation of the function evaluations across the processors. Reducing the over all computational is important in case the system to be used with time costly functions especially those that have many decision variables. The best speedup is for the ZDT6 problem. This is due to the nature of the problem as it requires more time to compute one function evaluation. The fixed population size speedup are acceptable and the best speedup is when 6 processors are used then starts to decrease due to communication overhead as the ZDT functions are not that expensive functions to compute.

V. Conclusions and Future Work

We have described a novel approach for approximating the Pareto front of multiobjective problems using a parallel evolutionary system. The system we propose integrates state-of-the-art EMOAs that run in a multi-start parallel approach. The engineered system was tested on standard benchmark problems and compared with well known EMOAs. The obtained

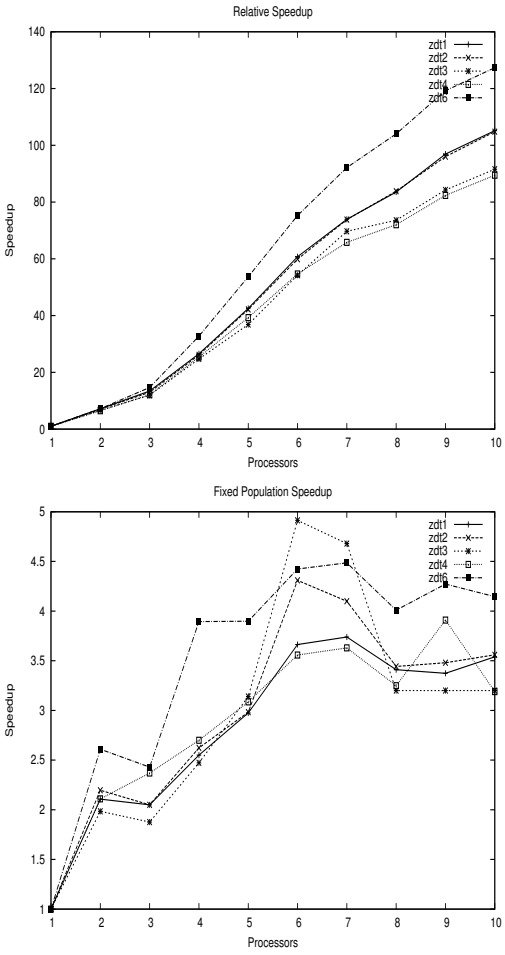

Fig. 4. Average relative speedup and with fixed population size for ZDT1, ZDT2, ZDT3, ZDT4 and ZDT6.

Pareto fronts gave very good results in terms of generational distance and hypervolume metrics. In fact, PESMO achieved new GD results for all the test problems and new hypervolume value for the difficult ZDT4. Also, de-activating the constraints is important for disconnected problems. The computational speedup was super linear due to the population size division on few processors.

To sum up, the work presented in this paper has the following novelties:

- We propose a novel parallel evolutionary system for defining and dividing the Pareto-front in the objective space of multi-objective problems.
- We introduce the use of intelligent starting points rather than randomly generated initial population. Each processor is given a population within the lower and upper boundaries of its sub-Pareto front.
- PESMO is adaptive as it determines at run-time which objective function should be used to divide the objective space.

For future work, possibly looking at how to make the system adaptive where processors can change the boundaries of their

partitions in order to load balance work according to a certain threshold. Also, specialised crossover and mutation operators that can be more suited to work in highly niche partitions.

REFERENCES

[1] J. Shi, Q. Zhang and J. Sun, PPLS/D: Parallel Pareto Local Search Based on Decomposition, IEEE Transactions on Cybernetics, vol. 50, no. 3, pp. 1060-1071, March 2020.

[2] N. Kantour, S. Bouroubi, D. Chaabane, parallel MOEA with criterion-based selection applied to the Knapsack Problem, Applied Soft Computing, 80, 2019, Pages 358-373.

[3] M. Hamdan. A dynamic polynomial mutation for evolutionary multi-objective optimization algorithms. International Journal on Artificial Intelligence Tools, 20(1), pp. 209-219, 2011.

[4] M. Hamdan, Revisiting the distribution index in simulated binary crossover operator for evolutionary multiobjective optimisation algorithms. In 2014 Fourth International Conference on Digital Information and Communication Technology and its Applications, Thailand, 2014, pp. 37-41.

[5] B. Mishra, S. Mishra, S. Singh. Parallel Multi-Criterion Genetic Algorithms: Review and Comprehensive Study. International Journal of Applied Evolutionary Computation (IJAEC), 7(1), 50-62,2016.

[6] E. Alba, F. Chicano, On the behavior of parallel genetic algorithms for optimal placement of antennae in telecommunications, International Journal of Foundations of Computer Science 16 (2) (2005) 343–359.

[7] E. Alba, J. M. Troya, A survey of parallel distributed genetic algorithms, Complexity 4 (4) (1999) 31–52.

[8] E. Alba, J. M. Troya, Analyzing synchronous and asynchronous parallel distributed genetic algorithms, Future Generation Computer Systems 17 (2001) 451–465.

[9] N. Beume, B. Naujoks, M. Emmerich, SMS–EMOA: Multiobjective selection based on dominated hypervolume, European Journal of Operational Research 127 (3) (2007) 1653–1669.

[10] N. Beume, G. Rudolph, Faster S-Metric Calculation by Considering Dominated Hypervolume as Klee's Measure Problem, in: B. Kovalerchuk (ed.), Proceedings of the Second IASTED Conference on Computational Intelligence, ACTA Press, Anaheim, 2006, pp. 231–236.

[11] J. Branke, H. Schmeck, K. Deb, M. Reddy, Parallelizing multi-objective evolutionary algorithms: Cone separation, in: IEEE Congress on Evolutionary Computation (CEC), 2004, pp. 1952–1957.

[12] C. A. Coello Coello, D. A. Van Veldhuizen, G. B. Lamont, Evolutionary Algorithms for Solving Multi-Objective Problems, Kluwer Academic Publishers, New York, 2002, iSBN 0-3064-6762-3.

[13] D. W. Corne, N. R. Jerram, J. D. Knowles, M. J. Oates, PESA-II: Region-based Selection in Evolutionary Multiobjective Optimization, in: L. Spector, E. D. Goodman, A. Wu, W. Langdon, H.-M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. H. Garzon, E. Burke (eds.), Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2001), Morgan Kaufmann Publishers, San Francisco, California, 2001, pp. 283–290.

[14] F. de Toro Negro, J. Ortega, E. Ros, S. Mota, B. Paechter, J. Martin, PSFGA: Parallel processing and evolutionary computation for multiobjective optimisation, Parallel Computing 30 (2004) 721–739.

[15] K. Deb, R. Agrawal, Simulated binary crossover for continous search space, Complex Systems 9 (1995) 115–148.

[16] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: Nsga-ii, IEEE Transactions on Evolutionary Computation 6 (2) (2002) 182–197.

[17] K. Deb, S. Tiwari, Omni-optimizer: A generic evolutionary algorithm for single and multi-objective optimization, European Journal of Operational Research 185.

[18] J. J. Durillo, A. J. Nebro, C. A. C. Coello, F. Luna, E. Alba, A comparative study of the effect of parameter scalability in multi-objective metaheuristics, in: IEEE Congress on Evolutionary Computing, Hong Kong, 2008.

[19] J. J. Durillo, A. J. Nebro, F. Luna, B. Dorronsoro, E. Alba, jMetal: A java framework for developing multi-objective optimization metaheuristics, Tech. Rep. ITI-2006-10, Departamento de Lenguajes y Ciencias de la Computación, University of Málaga, E.T.S.I. Informática, Campus de Teatinos (December 2006).

[20] A. E. Eiben, J. E. Smith, Introduction to Evolutionary Computing, 2nd ed., Springer, Natural Computing Series, 2007.

[21] D. E. Goldberg, Genetic Algorithms in Search, Optimization & Machine Learning, Addison-Wesley, Reading, MA, USA, 1989.

[22] N. Hansen, A. Ostermeier, Completely derandomized self-adaptation in evolution strategies, Evolutionary Computation 9 (2) (2001) 159–195.

[23] M. M. T. Hiroyasu, S. Watanabe, The new model of parallel genetic algorithms in multi-objective optimization problems-divided range multi-objective genetic algorithm, in: IEEE Congress on Evolutionary Computation, 2000, pp. 333–340.

[24] C. Igel, T. Glasmachers, V. Heidrich-Meisner, Shark, Journal of Machine Learning Research 9 (2008) 993–996.

[25] A. L. Jaimes, C. A. C. Coello, MRMOGA: a new parallel multi-objective evolutionary algorithm based on the use of multiple resolutions, Concurrency and Computation: Practice and Experience 19 (4) (2007) 397–441.

[26] J. D. Knowles, D. W. Corne, Approximating the Nondominated Front Using the Pareto Archived Evolution Strategy, Evolutionary Computation 8 (2) (2000) 149–172.

[27] J. Mehnen, T. Michelitsch, K. Schmitt, T. Kohlen, pMOHypEA: Parallel evolutionary multiobjective optimization using hypergraphs, Technical Report Reihe CI-189/04, University of Dortmund, iSSN 1433-3325 (2004).

[28] K. Miettinen, Non-linear Multiobjective Optimization, ser, vol. 12 of Kluwer's International Series in Operations Research and Management Science, Kluwer Academic Pppublishers, 1999.

[29] A. J. Nebro, J. J. Durillo, F. Luna, B. Dorronsoro, E. Alba, A cellular genetic algorithm for multiobjective optimization, in: D. A. Pelta, N. Krasnogor (eds.), Proceedings of the Workshop on Nature Inspired Cooperative Strategies for Optimization (NICSO 2006), Granada, Spain, 2006, pp. 25–36.

[30] A. J. Nebro, F. Luna, E. Alba, B. Dorronsoro, J. Durillo, A. Beham, AbYSS: Adapting scatter search to multiobjective optimization, IEEE Transcations on Evolutionary Computing 20.

[31] M. J. Quinn, Parallel Programming in C with MPI and OpenMP, McGraw Hill, New York, USA, 2003.

[32] W. Rivera, Scalable parallel genetic algorithms, Artificial Intelligence Review 16 (2001) 153–168.

[33] D. J. Sheskin, Handbook of Parametric and Nonparametric Statistical Procedures, 4th ed., CRC Press, 2007.

[34] S. N. Sivanandam, S. Sumathi, T. Hamsapriya, A hybrid parallel genetic algorithm approach for graph coloring, International Journal of Knoweldge-Based and Intelligent Engineering Systems 9 (3) (2005) 249–259.

[35] F. Streichert, H. Ulmer, A. Zell, Parallelization of multi-objective evolutionary algorithms using clustering algorithms, in: C. A. Coello Coello, A. H. Aguirre, E. Ziztler (eds.), Conference on Evolutionary Multi-Criterion Optimization (EMO 2005), vol. 3410 of LNCS, Guanajuato, Mexico, 2005, pp. 92–107.

[36] K. Tan, Y. Yang, C. Goh, A distributed cooperative coevolutionary algorithm for multiobjective optimization, Evolutionary Computation, IEEE Transactions on 10 (5) (2006) 527–549.

[37] D. A. V. Veldhuizen, G. B. Lamont, Multiobjective Evolutionary Algorithm Research: A History and Analysis, Tech. Rep. TR-98-03, Department of Electrical and Computer Engineering, Graduate School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, Ohio (1998).

[38] L. A. Wilson, M. D. Moore, Cross-pollinating parallel genetic algorithm for multi-objective search and optimization, International Journal of Foundations of Computer Science 16 (2) (2005) 261–280.

[39] Q. Zhang, H. Li, Moea/d: A multi-objective evolutionary algorithm based on decomposition, IEEE Trans. on Evolutionary Computation 11 (6) (2007) 712–731.

[40] E. Zitzler, K. Deb, L. Thiele, Comparison of multiobjective evolutionary algorithms: Embirical results, Evolutionary Computing 8 (2) (2000) 173–195.

[41] E. Zitzler, M. Laumanns, L. Thiele, Spea2: Improving the strength pareto evolutionary algorithm, Tech. Rep. 103, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH), Zurich, Switzerland (2001).

[42] E. Zitzler, L. Thiele, Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach, IEEE Transactions on Evolutionary Computation 3 (4) (1999) 257–271.

[43] J. Shi, Q. Zhang, B Derbel, A. Liefooghe, S. Verel Using Parallel Strategies to Speed up Pareto Local Search. In: Shi Y. et al. (eds) Simulated Evolution and Learning. SEAL 2017. Lecture Notes in Computer Science, vol 10593. Springer, Cham