Multiobjective Distinct Candidates Optimization (MODCO): A Cluster-Forming Differential Evolution Algorithm

Peter Dueholm Justesen[1], Rasmus K. Ursem[2]

[1] Department of Computer Science
University of Aarhus,
Åbogade 34, DK-8200 Århus N, Denmark.
juste@daimi.au.dk
[2] Department of Fluid Mechanics, R&T
Grundfos Management A/S
Poul Due Jensens Vej 7, DK-8850 Bjerringbro, Denmark

Abstract. Traditionally, Multiobjective Evolutionary Algorithms (MOEAs) aim at approximating the entire true pareto-front of their input problems. However, the actual number of solutions with different trade-offs between objectives in a resulting pareto-front is often too large to be applicable in practice. The new field Multiobjective Distinct Candidates Optimization (MODCO) research is concerned with the optimization of a low and user-defined number of clearly distinct candidates. This dramatically decreases the amount of post-processing needed in the decision making process of which solution to actually implement, as described in our related EMO 2009 paper "Multiobjective Distinct Candidates Optimization (MODCO): A new Branch of Multiobjective Optimization Research".

In this paper, we introduce the first algorithm designed for the challenges of MODCO; providing a given number of distinct solutions as close as possible to the true pareto-front. The algorithm is using subpopulations to enforce clusters of solutions, in such a way that the number of clusters formed can be set directly. The algorithm is based on the Differential Evolution for Multiobjective Optimization (DEMO) algorithm versions, but is exchanging the crowding/density measure with two alternating secondary fitness measures. Applying these measures ensures that subpopulations are attracted towards knee regions while also making them repel each other if they get too close to one another. This way subpopulations traverse different parts of the objective space while forming clusters each returning a single distinct solution.

Key words: Multiobjective Optimization, Evolutionary Algorithms, Differential Evolution, Distinct Candidates, Subpopulations, Clustering, NSGA-II, SPEA2, DEMO, MODCO

1 Introduction

Multiobjective optimization (MO) is the discipline of finding the optimal set of solutions of problems having several, usually conflicting objectives. In traditional algorithmic research on MO, algorithms are evaluated on the following three goals: The algorithm's ability to (1) find a set of solutions as close to the true pareto-front as possible, (2) ensure an even distribution of solutions on the pareto-front, and (3) have a high spread of solutions, i.e., to find extreme solutions for each of the involved objectives. In short, closeness, distribution, and spread. Although these goals are desirable from a theoretical point-of-view, they are not from a practical point-of-view. Post-processing several hundred alternative solutions returned by traditional MO algorithms is often impossible because time and money may disallow further analysis of many solutions using advanced simulations or physical prototypes.

To address this issue and a number of other practically motivated challenges, we introduce Multiobjective Distinct Candidates Optimization (MODCO) as a new branch of MO research. In short, MODCO defines three alternative goals to MO research. The ideal MODCO algorithm aims at (1) closeness to true pareto front, (2) return a user-defined number of distinct solutions, and (3) return solutions in knee regions. In short, (1) closeness, (2) global distinctiveness, and (3) local multiobjective optimality. Whereas this paper introduces the first MODCO algorithm, a more elaborate argumentation for the soundness of MODCO and why this should be considered a new branch of MO research is given in our related paper also submitted to EMO 2009 [9].

In this paper, we propose the Cluster-Forming Differential Evolution algorithm (CFDE), using subpopulations subject to Differential Evolution (DE). It features a user-defined number of candidate solutions (K_{NC}), a user-defined performance distinctiveness (K_{PD}), and the ability to converge to knee regions, see [9] for further details. Hence, this first MODCO algorithm does not allow user-defined design distinctiveness (K_{DD}) or incorporation of simulator accuracy (K_{SA}). In short, performance distinctiveness (K_{PD}), design distinctiveness (K_{DD}) and category distinctiveness (K_{CD}) allow the user to set how similar or different the few returned solutions should be wrt. performance (objective space), design (search space), and the MODCO problem's category functions (see [9]).

2 Cluster-Forming Differential Evolution

Differential Evolution (DE) was suggested by Price and Storn [4], and was initially designed for single-objective optimization. However, with the growing interest in solving multiobjective problems, it has been successfully adopted in multiobjective evolutionary algorithms. Recently, Robič and Filipič combined both NSGA-II and SPEA2 selection with the Differential Evolution scheme for solution reproduction. These alternate versions were named DEMO^{NSII} and DEMO^{SP2} and have been shown to outperform the more classic multiobjective genetic algorithms NSGA-II and SPEA2 [1–3, 5, 6]. The Cluster-Forming Differential Evolution (CFDE) algorithm is based on the DEMO-scheme, but is modified for cluster forming using subpopulations and two specialized secondary fitness measures.

2.1 Data structures and notation

The population in the CFDE algorithm is stored in a vector of vectors, with each entry in the vector defining a subpopulation. For the rest of this article, we assume WLOG that the user-defined number of distinct candidates K_{NC} divides the population size N, i.e. that $N \bmod K_{NC} == 0$. We further use a centroid vector of M-dimensional points, and a temporary offspring vector for migration.

In pseudocode, we let P_i refer to subpopulation i of the full population P, whereas $x_{i,j}$ refers to the individual at the j'th position in subpopulation i. For global ranking and truncation, we denote the current generation population P_t, and subpopulations $P_{i,t}$.

We access cluster centroids using C_i to denote the M-dimensional point in objective space corresponding to the centroid of subpopulation i, and $C_{i,m}$ to access the m'th entry in this. Lastly, we let $f(x)$ denote the objective vector of individual x, and $f_m(x)$ to denote the m'th entry of this.

2.2 Main algorithm

Pseudocode of the main algorithm can be seen in Algorithm 1, where we let $minDist(C_i)$ denote the function returning the minimum distance from centroid C_i to the nearest other centroid. Further, the calculation of σ is problem-dependent, as explained in [9], and further demonstrated later in this paper.

First, the CFDE algorithm performs global mating with replacement, as in usual DE. However, it stores the incomparable offspring in a temporary offspring vector, until it can be determined, which subpopulation they should belong too. From the parent part[3] of the subpopulations, a centroid for each is then calculated. Following this, the incomparable offspring are migrated to the subpopulations with the nearest centroid.

At this point, the CFDE algorithm determines which of the two secondary fitness measures to use for each subpopulation. The secondary fitness measure is each individual's distance to the nearest other centroid, if the subpopulations centroid is too close to its nearest neighboring centroid. In case the centroid of the subpopulation is sufficiently far away from its neighbours, the secondary fitness measure is Branke et al.'s utility function, favoring individuals in knee regions [8].

Still, CFDE maintain focus on convergence towards the true pareto-front, so it then assigns to each individual a *global* pareto rank using the NSGA-II non-dominated sorting. This is used for assigning each individual a final fitness,

[3] See figure 1.

Algorithm 1 Cluster-Forming Differential Evolution

```
Require: Population size $N$, $K_{NC}$, $K_{PD}$
Ensure: $K_{NC}$ different non-dominated individuals.
 1: Initialize $K_{NC}$ subpopulations with $N/K_{NC}$ random individuals in each
 2: while Halting criterion has not been met do
 3:   Perform global DE-based mating - store incomparable offspring
 4:   Calculate subpopulation centroids $C_i$
 5:   Migrate incomparable offspring to nearest subpopulation wrt. centroid
 6:   for All $P_i \in P$ do
 7:     if $minDist(C_i) < \sigma$ then
 8:       Assign nearest other centroid distance to each individual $x_{i,j} \in P_i$
 9:     else
10:       Assign knee utility function value to each individual $x_{i,j} \in P_i$
11:     end if
12:   end for
13:   Assign final fitness wrt. global pareto rank, then secondary fitness
14:   Truncate subpopulations wrt. final fitness
15: end while
16: Return $K_{NC}$ solutions, by returning the non-dominated solution closest to the
    subpopulation centroid from each subpopulation.
```

such that the final fitness incorporates both rank and secondary fitness measure in a total order.

Finally each subpopulation is truncated to the original size of N/K_{NC} using the truncation mechanism of NSGA-II. Here, the main differences are, that truncation is done locally in subpopulations, and that the subpopulations are truncated using one of the two secondary fitness measures, which is incorporated in the final fitness. Hence, some subpopulations may be truncated using distance, and others using the knee utility function. This way subpopulations may be attracted to different knee regions, while forming clusters during the evolutionary process. As we return only one solution from each subpopulation, we get the wanted number of distinct solutions returned.

2.3 Subpopulation based Differential Evolution

For subpopulation based DE, we will use the DE scheme with replacement, such that parents are replaced if they are dominated by their offspring, and offspring are discarded if they are dominated by their parents. However, the subpopulation approach makes it necessary to determine which subpopulation a newly created offspring should belong to, in case the offspring is incomparable, such that it neither dominates or is dominated by their parents.

In case an offspring is incomparable to its parent, it is stored in a temporary vector in order to enable migration after we have calculated subpopulation centroids. This ensures that incomparable offspring will always belong to the subpopulation they are closest to, and that the parent part of the subpopulation remains fixed. This is illustrated in figure 1, which shows the population after migration. Note that we denote the parent part as the N/K_{NC} first entries in the subpopulation vector, even though some of these entries may have been overwritten by dominating offspring candidates. This is natural, as these offspring contribute to mating as soon as they have replaced their parents.

The subpopulation based Differential Evolution algorithm is based on the $rand/1/bin$ standard DE scheme, and is depicted in Algorithm 2. As usual, we perform the algorithm for all individuals[4] $x_{i,j} \in P$, creating one offspring per parent, which is treated according to the cases described above.

Algorithm 2 Subpopulation based Differential Evolution

```
Require: Parent $x_{i,j}$, crossover factor $CF$, scaling factor $F$.
Ensure: Candidate $c_{i,j}$.
1: Randomly select three individuals $x_{i_1,j_1}$, $x_{i_2,j_2}$, $x_{i_3,j_3}$ from $P$, where $x_{i,j}$, $x_{i_1,j_1}$,
   $x_{i_2,j_2}$ and $x_{i_3,j_3}$ are pairwise different.
2: Calculate candidate $c_{i,j}$ as $c = x_{i_1,j_1} + F \cdot (x_{i_2,j_2} - x_{i_3,j_3})$.
3: Modify candidate $c_{i,j}$ by binary crossover with the parent $x_{i,j}$ using $CF$.
```

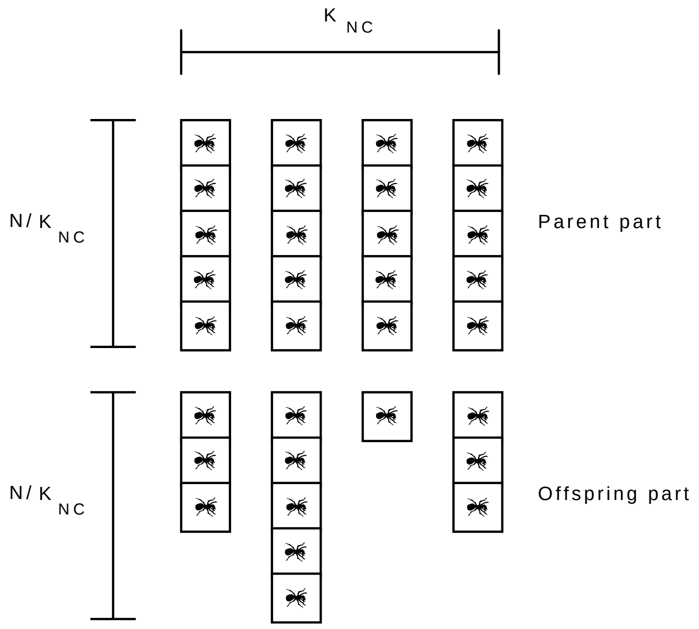

Fig. 1. Population after migration

[4] which at this point is only the parent part, see figure 1.

2.4 Calculating subpopulation centroids

The simple subpopulation centroid calculation is a measure of the center of the current elite part of the subpopulation, namely the parent part, as depicted in figure 1 and described above. This part consists of the elite from the previous generation, mixed with the offspring, which replaced their parents. These individuals are the current best, and therefore we can use this fixed size part of the subpopulation to define a centroid. For each subpopulation P_i, we calculate the centroid $C_i = [C_{i,1}, C_{i,2}...C_{i,M}]$ as the average point of the elite in objective space:

$$C_{i,m} = \frac{\sum_{j=1}^{N/K_{NC}} f_m(x_{i,j})}{N/K_{NC}}, m = 1..M. \tag{1}$$

2.5 Migration

All incomparable offspring are at this point in a temporary offspring vector. To migrate these offspring, the algorithm iterates through the offspring vector, and move each individual to the subpopulation, whose centroid is the closest to the individual in objective space. More formally, this is done by appending the individual to the subpopulation, while removing it from the offspring vector. Note, that we do not allow duplicates within subpopulations.

After this migration, the full population is present, which is depicted in figure 1, but it may be noted that some subpopulations may increase their offspring part size beyond N/K_{NC}. This, however, is a theoretical possibility, and most often subpopulation sizes will vary from N/K_{NC} to $2N/K_{NC}$.

2.6 Secondary fitness assignment

Having calculated the subpopulation centroids, we can now use this to assign a secondary fitness to individuals, depending on whether subpopulations are too close to each other or not, wrt σ. So we may assign different secondary measures to subpopulations. However, this is no problem with regard to truncation, which also happens within subpopulations, so if two individuals are not judged by the same secondary measure, they will not be compared during truncation. The measures proposed are the utility-function proposed by Branke et al. [8], which focus search on knee regions, and a new centroid distance measure, which makes subpopulations repel each other.

Using utility function The utility function proposed in [8], is intended to discover knee regions by calculating an average fitness value for a large number of randomly sampled weight vectors. If this average fitness is good, the individual is more likely to reside in a knee-region. Knee-regions are characterized by the fact that a small improvement in one objective, will result in large deterioration in another objective.

The utility function takes only one argument, *precision*, denoting the number of sample weight vectors to apply. Let λ denote the weight vector of dimension M, with $\sum_m \lambda_m = 1$. Then we calculate the secondary fitness SF with precision *precision* of each individual $x \in P_i$ as:

$$SF(x) = \frac{\sum_{p=1}^{precision} \lambda_p \cdot f(x)}{precision} \tag{2}$$

Using centroid distance Having used migration, incomparable offspring are at this point distributed to subpopulations, as seen in figure 1. Therefore we can go through the full population, and assign to each individual the distance in objective space, to the nearest other subpopulation centroid. Now, let $dist(x, y)$ denote the distance in objective space between point x and point y, each of dimension M. Further, let $min(S)$ denote the minimal element of the set S.

For the subpopulation P_i, we may then assign to each individual $x \in P_i$ a secondary fitness SF as:

$$SF(x) = min(\{dist(f(x), C_j), j = 1..K_{NC}, j \neq i\}) \tag{3}$$

2.7 Assigning global pareto rank

The global pareto rank is intended to guide the search towards the best individuals wrt. dominance. Two very popular ways of assigning a pareto rank are based on the non-dominated sorting from NSGA-II or the strength pareto approach of SPEA2. These ranks can be viewed as a raw fitness, and are traditionally complemented by a crowding/density measure. Therefore, it is straight forward to exchange the original measures with the secondary fitnesses introduced above. We show how to do this using the approach of NSGA-II, but it is also possible to use the SPEA2 scheme for ranking and truncating.

The elitism mechanism in NSGA-II is based on non-dominated sorting of the current population P_t. For each individual $\in P_t$, non-dominated sorting assigns a non-dominated rank equal to the non-dominated front label, which is used to group the individuals into fronts. Here, the first front[5] consists of the populations non-dominated solutions, the next front consists of solutions dominated only by the first front and so on. Different ways of performing non-dominated sorting are described in [2].

NSGA-II further uses a crowding measure intended to be maximized, such that the higher the value, the better the individual. Assuming a maximization problem, we are also interested in maximizing our secondary fitness, such that we may directly replace the crowding measure with our assigned secondary fitness. This way we can utilize the truncation mechanism from NSGA-II directly, since we now have the two values needed for sorting, namely a rank to be minimized and a secondary fitness to be maximized. Note that for minimization objectives, we need to multiply the corresponding weights of Branke's knee utility function with -1, whereas the centroid distance always is to be maximized.

[5] Where all individuals are assigned rank 1.

2.8 Truncation

We can now easily apply the truncation mechanism of NSGA-II, with the major difference being that the procedure is now performed on each subpopulation $P_{i,t} \in P_t$ in generation t instead of on the full population. Truncation ensures that the size of $P_{i,t+1}$ is exactly N/K_{NC}.

After ranking, as many fronts as possible are accommodated in the next generation subpopulation $P_{i,t+1}$, where we include the lowest ranked fronts first. The last front to be inserted is sorted wrt. the secondary fitness measure, and only the best individuals wrt. this are chosen for inclusion. This truncation mechanism favors rank first, and a good secondary fitness value next.

3 Experimental setup and results

In this section, we demonstrate the CFDE algorithm on 3 kinds of problems; the 2D ZDT problems [2, 1], the knee problems of Branke et al. [8], and all non-constrained 3D DTLZ problems [7], all with problem settings as suggested in the respective papers. We do this with respect to the MODCO goals:

- Convergence performance (MODCO goal 1)
- Global distinctiveness (MODCO goal 2)
- User-defined performance distinctiveness (MODCO goal 2)
- Local multiobjective optimality (MODCO goal 3)

Using this taxonomy, we address the different issues of CFDE usage according to the MODCO goals. First, we check CFDE convergence against the DEMO versions, which have demonstrated good performance on many problems [5, 6]. Next, we want to demonstrate convergence to K_{NC} clusters, how we may change solution diversity by setting K_{PD}, and finally that CFDE are able to locate knees.

For reference, we here provide an overview of the parameters used for the experiments. For all experiments performed with the DEMO algorithm versions, we have used a population size $N = 100$, and a DE setting with $F = 0.5$ and $CF = 0.3$, as used in [5, 6]. Number of generations used by DEMO for the test problems is listed in Table 1.

Table 1. Settings for DEMO^{NSII} and DEMO^{SP2}

Problem	ZDT1	ZDT2	ZDT3	ZDT4	ZDT6	DO2DK	DEB2DK	DEB3DK
Generations	250	250	250	500	500	200	200	200
Problem	DTLZ1	DTLZ2	DTLZ3	DTLZ4	DTLZ5	DTLZ6	DTLZ7	
Generations	300	300	500	500	200	500	200	

In table 2 we see the settings for the CFDE runs. As we will see, even with a lower number of generations, CFDE is very competitive to the DEMO versions on almost the entire test suite. For all runs except one, we have used $N = 100$, and the same DE settings as for DEMO; $F = 0.5$ and $CF = 0.3$. The exception is DTLZ3, were we have used a population size of 160 individuals, to match $K_{NC} = 8$. This way, we use approximately as many evaluations on this problem as DEMO, as we perform 300 generations for CFDE[6]. We demonstrate the use of K_{PD} only on knee problems, as this is only relevant for such problems. For ZDT and DTLZ problems, we therefore always set $\sigma = \infty$, effectively disabling knee search. For DTLZ problems, we have used a higher K_{NC} to ensure that we find both extreme and intermediate trade-offs.

Table 2. Settings for CFDE

Problem	ZDT1	ZDT2	ZDT3	ZDT4	ZDT6	DO2DK	DEB2DK	DEB3DK
Generations	100	100	100	500	500	200	200	200
K_{NC}	5	5	5	5	5	5	4	5
Problem	DTLZ1	DTLZ2	DTLZ3	DTLZ4	DTLZ5	DTLZ6	DTLZ7	
Generations	300	300	300	200	200	500	200	
K_{NC}	10	10	8	10	10	10	4	

3.1 Convergence performance

To deal with the different cardinality of more standard MOEAs and the CFDE algorithm, we use the universal notion of dominance. Here, we compare the CFDE algorithm against DEMO^{NSII} and DEMO^{SP2}. One algorithmic argument for MODCO is that the low number of returned solutions allows a more focused search because MODCO does not aim at an even distribution, see [9] for details. Consequently, a MODCO algorithm should be able to return solutions closer to the true pareto front.

We wish to investigate in which extent the returned solutions from the CFDE algorithm dominate the most similar solutions from the returned population of the competing MOEAs, where similarity is measured as distance in objective space. This way we see if the CFDE approach is competitive to simply picking K_{NC} solutions from the resulting populations of the DEMO versions.

For all results below, these are generated using the NSGA-II version of global ranking and truncation in the CFDE algorithm. We have used 20 runs for both the DEMO versions and for CFDE on each problem. For each generated population of CFDE, we have compared each resulting individual to its most similar counterpart from each of the DEMO populations. This gives a percentage of the amount of dominating, dominated and incomparable individuals CFDE was able to produce and is independent of K_{NC} and K_{PD}. Using $K_{NC} = 5$ we thereby get $5 \cdot 20 \cdot 20 = 2000$ comparisons for the problem, while $K_{NC} = 10$ yields 4000 comparisons.

[6] 160 individuals $\cdot$ 300 iterations = 48000 evaluations for CFDE vs. 50000 for DEMO.

Table 3. CFDE versus DEMO^{NSII} and DEMO^{SP2}

CFDE vs. DEMO(NSII)	ZDT1	ZDT2	ZDT3	ZDT4	ZDT6	DO2DK	DEB2DK	DEB3DK
Dominates	0	0	0.0995	0.7700	0	0.4680	0.8718	0.4455
Dominated	0	0	0	0	0	0	0	0
Incomparable	1	1	0.9005	0.2300	1	0.5320	0.1282	0.5545
CFDE vs. DEMO(SP2)								
Dominates	0	0	0.0810	0.6800	0.0100	0.2265	0.9081	0.3710
Dominated	0	0	0.0005	0	0.0400	0.0030	0	0
Incomparable	1	1	0.9185	0.3200	0.9500	0.7705	0.0919	0.6290

As can be seen from Table 3, CFDE is performing extremely well on the 2D ZDT problems and on the more simple knee problems. It is clear, that the compared DEMO individuals almost never dominate the CFDE individuals, while CFDE often produce individuals dominating their DEMO counterpart. For the more simple problems ZDT1 and ZDT2, all algorithms find optimal solutions and are therefore incomparable. For ZDT3, CFDE seems to outperform both DEMO versions, even with only 100 iterations. This may be caused by the more focused search around subpopulation centroids performed by CFDE during the run. Even for the much more difficult problem ZDT4 with many local fronts, CFDE seems to dominate again maybe caused by the more focused search not spending effort on equally distributing individuals. However, for ZDT6 with a low density of solutions near the true pareto-front, DEMO shows equal performance to CFDE.

Table 4. CFDE versus DEMO^{NSII} and DEMO^{SP2}

CFDE vs. DEMO(NSII)	DTLZ1	DTLZ2	DTLZ3	DTLZ4	DTLZ5	DTLZ6	DTLZ7
Dominates	0.1432	0.3085	0.3112	0.3210	0.3580	0	0.0493
Dominated	0	0	0	0	0	0.2000	0
Incomparable	0.8568	0.6915	0.6888	0.6790	0.6420	0.8000	0.9507
CFDE vs. DEMO(SP2)							
Dominates	0.1740	0.1600	0.2960	0.3765	0.3255	0.0000	0.0781
Dominated	0	0	0.0300	0	0	0.1750	0
Incomparable	0.8260	0.8400	0.6740	0.6235	0.6745	0.8250	0.9219

In Table 4, it is clear that on DTLZ1 and DTLZ2, CFDE outperforms both DEMO version with an equal number of evaluations. This is however more clear for DEMO^{NSII} than for DEMO^{SP2}. For DTLZ3, CFDE outperforms both DEMO versions, using a higher populations size, but with a lower number of iterations resulting in a similar number of evaluations as noted before. For this problem, CFDE needed a higher diversity than normal to perform well due to the heavy complexity of the problem. On DTLZ4, CFDE outperforms DEMO with less than half the number of iterations, most likely due to the optimal front being a flat curve easy to attain. This also goes for DTLZ5, but with an equal number of evaluations performed. Only on DTLZ6, the harder version of DTLZ5, DEMO outperforms CFDE. This problem is also hard for more traditional MOEAs as noted in [7]. For DTLZ7, we have used $K_{NC} = 4$, as there are 4 optimal planes, which are all being attained by CFDE which on this problem demonstrates only a slightly superior performance.

Overall, CFDE outperforms or has same performance as the two DEMO versions on all test problems except DTLZ6. So CFDE appears competitive to the DEMO versions, even if we have only compared the most similar solutions. More elaborate measures for MOEAs with different result cardinality is future works.

3.2 Global distinctiveness

Global distinctiveness is achieved by the CFDE algorithm using the centroid distance to repel subpopulations. Figure 2 and 3 displays the returned results of 20 runs of the CFDE algorithm on ZDT1 and ZDT3, using $K_{NC} = 5$ and $K_{NC} = 10$. As mentioned above, we here set $\sigma = \infty$. Similar robust convergence is seen for the other test problems, i.e., CFDE found roughly the same set of distinct candidates in repeated runs.

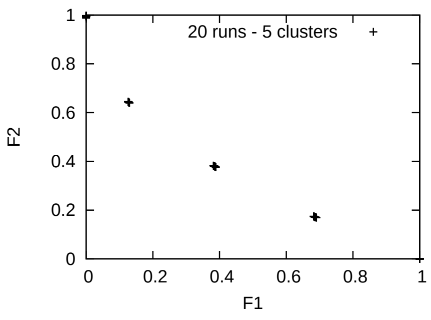

Fig. 2. ZDT1 plot - 20 runs - 5 clusters

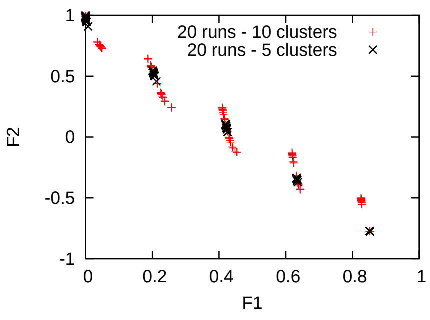

Fig. 3. ZDT3 plot - 20 runs - 5/10 clusters

As can be seen in figure 2, all of the 20 runs of CFDE returned similar distinct solutions. In figure 3, we see that using $K_{NC} = 5$ ensures a result returned from each of the 5 patches of the true pareto-front and again we see only a small variation. However, using $K_{NC} = 10$ makes the returned results be much more spread in the 20 runs, since there are now more clusters to be formed than there are discontinuous patches. As can be observed from the density, solutions will here most often seek the most outer part of the patches making the returned solutions as distinct as possible.

3.3 User-defined performance distinctiveness

The MODCO parameter $K_{PD} \in [0,1]$ allows the DM to set how distinct the returned solutions should be. A low value corresponds to a low distinctiveness and a high value to a high distinctiveness. See [9] for further information.

To demonstrate the effects of changing K_{PD}, we have chosen DEB3DK as an illustrative example. DEB3DK is interesting with its 3 objectives and single knee, since this allow us to visualize the effect of altering the balance between knee search and subpopulation repelling in 3D. We have used DEB3DK with only one knee, i.e. $K = 1$.

We first demonstrate the calculation of σ used in the CFDE algorithm. First, we will assume settings $K_{PD} = 1$ and $K_{NC} = 5$. For DEB3DK, we may use reference points $z^{**} = (0,0,0)$ and $z^I = (8,8,8)$, which spans the interesting part of the objective space. Then we can calculate:

$$\sigma = K_{PD}/K_{NC} \cdot ||z^{**} - z^I|| = 1/5 \cdot \sqrt{192} \approx 2.77 \tag{4}$$

In this settting, subpopulations will repel each other if they get within a distance of 2.77 of each others centroids. Setting $K_{PD} = 1$ should ensure maximum global distinctiveness, such that we get clusters uniformly spread across the objective space spanned by the reference points. Contrary, setting K_{PD} close to zero[7] enables clusters to get closer to each other while searching for knees.

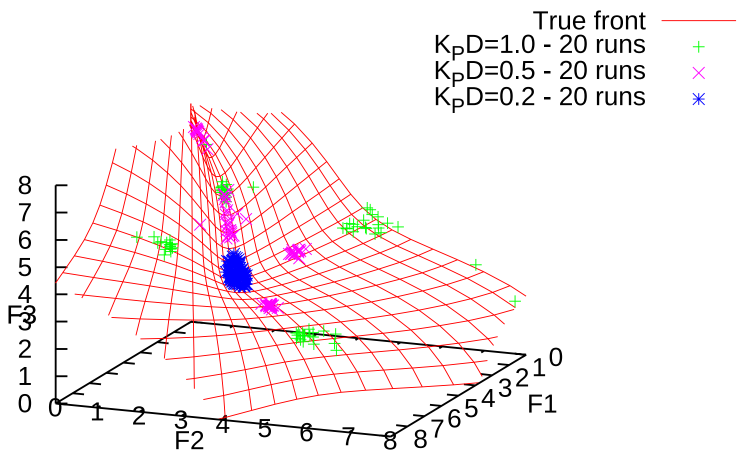

Fig. 4. DEB3DK plot - investigating user defined performance distinctiveness

Fgure 4 illustrates the results of setting K_{PD} to 0.2, 0.5 and 1.0. For $K_{PD} = 1$, the 5 clusters are equidistant around the single knee, where one cluster is placed. The four clusters not in the knee are repelled from each other as they

[7] but above, since $K_{PD} = 0$ allows for subpopulations to overlap.

reach a distance of 2.77 between centroids, as was demonstrated in the example calculation above. Interestingly, the four clusters not in the knee are located in the partial knees on the lines forming a cross. For $K_{PD} = 0.5$, we always hit the knee with one cluster. Further, it can be seen that some solutions has found other knee regions, crawling towards the one in the middle, but these are still not allowed to get too close to each other. Setting $K_{PD} = 0.2$ results in all clusters getting very close to the single knee region. Overall, it is clear that increasing K_{PD} indeed makes clusters repel each other more.

3.4 Local multiobjective optimality

Figure 4 and 5 illustrates the knee problems DO2DK and DEB2DK, with the resulting CFDE individuals of 20 runs. In the DO2DK problem, we set $K = 4$ and $s = 1.0$, such that we have exactly the same settings as has been used for creating the results illustrated in figure 4 in [8]. For the 20 runs depicted in figure 4, it may be noticed, that the density of solutions near knee regions is very high. When using $K_{NC} = 5$, CFDE finds the 4 knee regions very precisely, while one cluster typically hits an outer solution, or is caught in-between knee regions. For DO2DK, we have used $K_{PD} = 0.75$ corresponding to $\sigma = 1.5$. This way we keep clusters separated, while still allowing for knees to be found.

For the DEB2DK problem, we have used $K = 4$ to replicate the results illustrated in figure 5 in [8]. In figure 5, we again see that for the 20 runs the density of solutions near knees are very high. Here, we have to set K_{NC} to be equal to the number of knees, and it is clear that all knees are discovered in all runs. Here, we have used $K_{PD} = 0.2$ corresponding to $\sigma = 0.5$. This is low, so the centroid distance assignment is rarely used. Hence, subpopulations converge to knees, and as long as $\sigma > 0$, the clusters formed will not overlap.

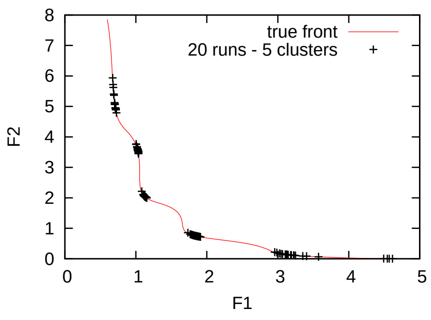

Fig. 5. DO2DK plot - 20 runs - 5 clusters - 4 knees

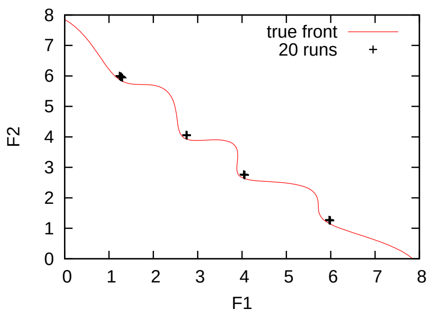

Fig. 6. DEB2DK plot - 20 runs - 4 clusters - 4 knees

From the figures above, and further figure 4, it is clear, that CFDE is indeed able to locate knee regions. Further, it has been demonstrated how to balance the search using K_{PD} resulting in different σ values.

3.5 Further aspects

In this section we discuss two further aspects of the CFDE algorithm, namely initialization in two dimensions, and an approach for constrained problems.

Initialization When initializing the first population, it is possible to make an easy initial clustering for problems with 2 objectives. We simply sort the initial random population based on the individuals first objective, before these are inserted into subpopulations. This way, less effort is spend on forming clusters during start of the run, enhancing efficiency. The approach has been used in all tests on 2D problems. The approach is only applicable for 2-dimensional problems, as for $M > 2$, there can be established no total order of individuals based on their objective vector.

Constrained problems Constrained problems require further comparison when applying selection. We recommend the GDE3 approach for such problems [10], such that individuals with no constraint violation will always be preferred over individuals who do violate constraints. Further, individuals with a lower constraint violation will be preferred over individuals with higher constraint violation. This will guide the search towards non-violated individuals, not loosing the property of being able to guide the search across constrained areas in objective space.

4 Conclusion

In this paper, we introduce the first MODCO algorithm called CFDE. Using subpopulation based Differential Evolution, the proposed CFDE algorithm allows the DM to specify the number of solutions returned, and how distinct they should be. The algorithm forms the desired number of clusters during the run using using two alternating secondary fitness measures assigned locally. These guides the search according to the goals of MODCO [9].

To conclude on this paper, we will again use the taxonomy presented in the start of the results section, Section 3, according to the MODCO goals:

Convergence CFDE clearly outperforms DEMO^{NSII} and DEMO^{SP2} when comparing distinct individuals on a broad test suite with ZDT, DTLZ and knee based problems. The reason for this is most likely the more focused search giving better convergence by allowing individuals to get very close within subpopulations. We give this as the most likely reason, as the approach resembles local search based exploitation having previously demonstrated good results e.g. in [11].

Global distinctiveness Regarding global distinctiveness, CFDE shows robust behavior by converging to the same set of distinct candidates on many independent runs. On test problems with no knees it was shown, that returned results of CFDE do try to achieve maximum global distinctiveness, but that this is naturally also depending on the number of subpopulations in use.

User-defined distinctiveness Further, our parameter investigations on K_{PD} shows that different settings of K_{PD} allows a user-controlled performance distinctiveness. Here in the form of distance-based performance distinctiveness [9].

Local multiobjective optimality As for local multiobjective optimality, our experiments have shown that the CFDE algorithm is able to locate knees even for changing parameter settings. This is important because we typically have no a priori knowledge on the structure of the problem's objective space.

Overall, the CFDE algorithm seems to comply very well with the goals of MODCO, and seems to be a very good alternative to post processing the results of more traditional MOEAs in order to decrease cardinality.

References

1. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE transactions on Evolutionary Computation 6, 2002, pp. 182 - 197.
2. Deb, K.: Multi-Objective Optimization using Evolutionary Algorithms. Wiley, 2002.
3. Zitzler, E., Laumanns, M. Thiele, L.: SPEA2: Improving the strength pareto evolutionary algorithm. Technical Report 103, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH) Zurich, Gloriastrasse 35, CH-8092 Zurich, Switzerland (2001).
4. Price, K. V., Storn, R.: Differential Evolution - a simple evolution strategy for fast optimization. Dr. Dobb's journal 22, 1997, pp. 18 - 24.
5. Robič, T., Filipič, B.: DEMO: Differential Evolution for Multiobjective Optimization. EMO 2005, LNCS 3410, pp. 520 - 533.
6. Robič, T., Filipič, B.: Differential Evolution Versus Genetic Algorithms in Multiobjective Optimization. In: LNCS volume 4403/2007, pp. 257 - 271.
7. Deb, K., Thiele, L., Laumanns M., Zitzler, E.: Scalable Test Problems for Evolutionary Multi-Objective Optimization. In: Proceedings of the 2002 Congress on Evolutionary Computation, CEC '02, pp. 825 - 830.
8. Branke, J., Deb, K., Dierolf H, Osswald, M.: Finding Knees in Multi-objective Optimization. In: LNCS volume 3242/2004, pp. 722-731.
9. Ursem, R.K., Justesen, P.D.: Multiobjective Distinct Candidates Optimization (MODCO) – A new Branch of Multiobjective Optimization Research. In submission to Evolutionary Multiobjective Optimization (EMO 2009). Preliminary download: `www.daimi.au.dk/~ursem/publications/Ursem_EMO2009_MODCO.pdf`
10. Kukkonen S., Lampinen, J.: GDE3: The third Evolution Step of Generalized differential evolution. In: Proceedings of the 2005 Congress on Evolutionary Computation, CEC '05, pp. 443 - 450.
11. Karthis, S., Deb, K., Miettinen: A Local Search Based Evolutionary Multiobjective Optimization for Fast and Accurate Convergence. In: LNCS volume 5199/2008, pp. 815 - 824.