Multiobjective Distinct Candidates Optimization: Locating a Few Highly Different Solutions in a Circuit Component Sizing Problem

Rasmus K. Ursem, Peter Dueholm Justesen

Structural & Fluid Mechanics, Grundfos Management A/S, Poul Due Jensens Vej 7
DK-8850 Bjerringbro, Denmark

Abstract

Traditional multi-objective optimization algorithms typically return several hundred non-dominated solutions. From practical point of view, a small set of 5–10 distinct candidates is often preferred because post-processing many solutions may be too costly, too time-consuming, or it may be too difficult to compare design differences.

In this paper, we introduce Multi-objective Distinct Candidates Optimization (MODCO) as an approach to find a user-defined low number of clearly different solutions wrt. performance and design. To demonstrate the potential of the MODCO approach, we suggest the General Cluster-Forming Differential Evolution (GCFDE) algorithm and test it on five well-known mechanical engineering problems and a new five-objective constrained problem from electrical engineering – the circuit component sizing problem of the Alpha Pro pump.

The experiments showed that GCFDE outperformed the compared algorithms on all six problems. Furthermore, our algorithm was able to return result sets in accordance with the user's settings for performance and design distinctiveness, which are two MODCO parameters controlling the result set.

Keywords: Multi-objective optimization, many-objective optimization, decision making, distinct candidates, MCDM, diversity management.

Email addresses: ursem@cs.au.dk (Rasmus K. Ursem), juste@cs.au.dk (Peter Dueholm Justesen)

1. Introduction

Successful application of multi-objective optimization to a real-world problem typically consists of two steps. First, the optimization step where the problem is set up, the chosen algorithm is executed, and all non-dominated solutions are gathered. Second, the decision making step where the single solution to implement is chosen among the non-dominated solutions found in step 1, see Figure 1 and Deb [1, pp. 5].

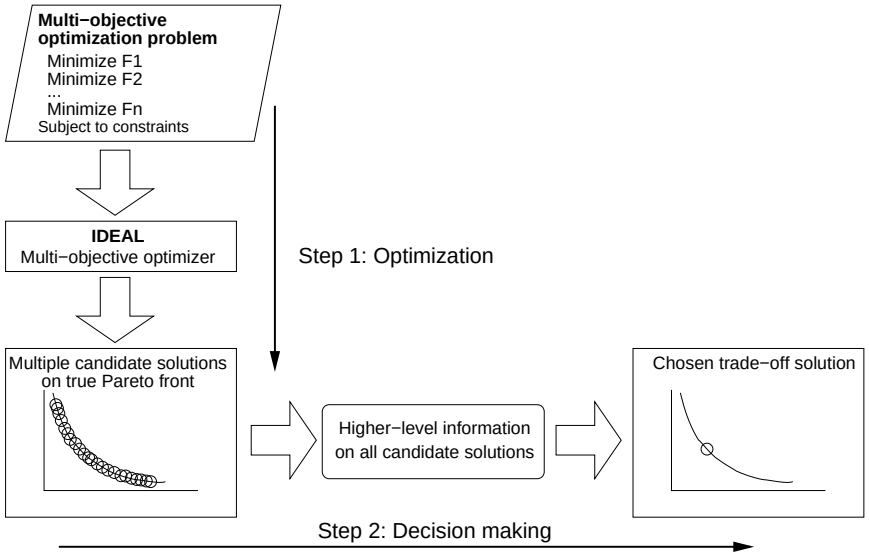


Figure 1: Process for application of multi-objective optimization.

In this process, the decision maker (DM) has to apply his preferences among the objectives to select the final solution. Veldhuizen and Lamont categorize the point in the process where the DM applies his preferences into three categories; 1) *a priori* – before the optimization is initiated, 2) *progressive* – during the optimization, and 3) *a posteriori* – after the optimization is finished [2]. Algorithms in category 1 typically transform the multi-objective problem into a single objective by specifying a utility function combining the multiple objectives. The weighted sum approach is the most widely known algorithm in this category. The progressive algorithms in category 2 usually incorporate the DM's preferences in the form of decision support systems, see [3] for a survey. Finally, category 3 algorithms exclude the DM's preferences from the search. Instead, they typically produce a large set of Pareto-optimal solutions for the DM to choose from in step 2.

The drawback of approaches 1 and 2 is that the DM has to make a choice regarding the importance of the involved objectives *prior* to the actual optimization, which may be difficult before the DM has seen any solutions. In addition, such choices are highly domain-specific and problem dependent, and algorithms are thus hard to generalize for a broader range of applications. In contrast, the traditional MO algorithms in category 3 are generally applicable. However, these algorithms produce hundreds or thousands of solutions and leave it to the DM to gather the "higher-level information" in step 2 on this set and choose the actual solution to implement. The often large set returned by *a posteriori* algorithms pose a serious problem because it may be *impossible* to gather "higher-level information" on such a large set. In short, time, money, and other reasons may prevent the application of the higher-level information gathering methods (further simulation, prototype construction, testing, etc.) on a set of more than 5–10 solutions. Consequently, we consider the current algorithms as either too domain-specific (category 1 or 2) or too general (category 3), because a huge set of candidate solutions is returned. Naturally, pruning the set using the DM's preferences is the obvious remedy for this drawback. However, this approach poses another problem because it may be difficult for the DM to state his preferences as explicit decision making rules. In our view, selection from a huge set or pruning the set tend to make the DM focus on the performance (objective space) and neglect the design differences (search space). In contrast, only investigating a few solutions promotes a better balance between the two. Furthermore, the low number of solutions allows the DM to apply preferences, use decision rules, and evaluate objectives not stated explicitly.

The MODCO approach addresses these challenges by incorporating *generalized preferences* with the goal of finding a small set of 5–10 distinct candidates to make step 2 manageable *without* stating explicit preferences. In MODCO, the concept *generalized preferences* covers a *priori* considerations that are relevant to most if not all real-world applications. This analysis answers the following questions (further elaborated in Section 2 and 3):

1. **Number of candidates:** $K_{NC} \in [1 : \infty] \subseteq \mathbb{N}$
   How many candidates is it practically and economically feasible to inspect, analyze, and compare in post-processing?
2. **Performance distinctiveness:** $K_{PD} \in [0 : 1] \subset \mathbb{R}$ or $[0 : 1]^M \subset \mathbb{R}^M$
   How different should the candidates be performance-wise? Is it an overall distinctiveness or on certain objectives?

3. **Design distinctiveness:** $K_{DD} \in [0:1] \subset \mathbb{R}$
   How different should the candidates be design-wise?
4. **Simulator accuracy:** $K_{SA} \in [0:1]^M \subset \mathbb{R}^M$
   What is the accuracy of the involved simulators?

In MODCO, the parameters K_{NC}, K_{PD}, K_{DD}, and K_{SA} constitute the generalized preferences, and they may be implemented as the secondary selection criterion in an algorithm. Hence, MODCO algorithms aim at reducing the DM's task in step 2 by dividing the higher-level information into two groups, *generalized preferences* as an a priori analysis to step 1 and the *domain-specific information gathering* as a precursor to the decision making in step 2. In this approach, the domain-specific information gathering includes further investigations such as visual inspection, detailed simulation, and prototype testing on the distinct candidates followed by evaluation of the DM's implicit or explicit preferences regarding the objectives. Thus, a MODCO algorithm combines categories 1 and 3 by integrating the generalized preferences a *priori* and leaving the domain-specific part to a *manageable* second step a *posteriori*.

In contrast to methods with explicitly stated preferences, the MODCO approach allows incorporation of rather vague statements from the DM or domain expert. For example, a domain expert may say *"For this problem, I know that many somewhat different solutions have roughly the same performance."* In MODCO, such a statement can be transformed into $K_{PD} = 0.0$ meaning "roughly same performance" and $K_{DD} = 1.0$ or perhaps $K_{DD} = 0.5$ representing a desire for highly or somewhat different solutions.

In relation to published research, a simple classification would be to categorize multi-criterion decision making (MCDM) research as producing category 1 and 2 optimization algorithms, and evolutionary MO research as introducing category 3 algorithms. In specific relation to the MODCO approach, the "modeling to generate alternatives (MGA)" suggested by Brill [4] address the challenge of handling non-modeled/implicit objectives by finding a small set of distinct candidates to present to the DM (for applications see [5, 6, 7, 8]). A more elaborate survey is provided in Section 4 as it is necessary to describe the MODCO ideas before related research can be discussed.

The paper is structured as follows. Section 2 provides the motivation for the MODCO approach by summarizing 6 years of observations from real-world industrial MO problems. Section 3 lists the features of the ideal MODCO algorithm and the goals of the MODCO approach. After having

introduced the MODCO ideas, we provide a survey of related research in Section 4. Section 5 introduces our novel MODCO algorithm. In Section, 6 we demonstrate its usefulness on a set of well-known mechanical engineering problems and a real-world circuit component sizing problem provided by Grundfos. Finally, Section 7 concludes the paper.

2. Motivation for MODCO algorithms

The application of multi-objective optimization in an industrial context raises a number of interesting challenges, dilemmas, and unforeseen obstacles. The following observations are gathered from more than 6 years of work at Grundfos R&T solving more than 30 multi-objective optimization tasks in very diverse engineering disciplines including fluid mechanics, motor design, motor control, structural mechanics, electronics, robust design optimization, value-chain optimization, production optimization, etc. Discussions with the DMs involved in these projects have lead to a number of considerations that form the basis for the MODCO approach. In short, the arguments for incorporating generalized preferences into the optimization algorithm fall in three categories:

1. Post-processing many Pareto-optimal solutions.
2. Physical realization of a solution.
3. Decision making among large sets of solutions.

These categories are discussed and elaborated in the following subsections.

2.1. Post-processing many Pareto-optimal solutions

In an industrial application context, multi-objective optimization is often only a small step in a large design process. Hence, the solutions found by an MO algorithm need to be further investigated to verify each solution against the results from other steps of the design process. Consequently, post-processing a large number of solutions is typically infeasible for the following reasons: i) It is too expensive, ii) it is too time-consuming, and iii) the optimization problem may only cover a part of the full design.

Regarding economy, it is often rather expensive to construct a physical prototype of a simulated design. For instance, a 3D prototype print and performance test of a single pump housing may cost up to 5000 Euro. Thus, investigating 100 candidates is out of the question.

Concerning time, a large set of candidate solutions is also problematic if it takes a lot of time to post-process a single solution. As mentioned above, the optimization is often only a small step in a larger design process and further extended simulation may be needed to verify the design in greater detail. For example, conducting a full-range CFD simulation on a design may take several days or weeks, which makes post-processing of large sets impossible. Furthermore, such an investigation often only constitutes a part of the simulations carried out on a design. Other types of analyses include stress analysis, cost calculation, robust design investigations, etc.

In addition to economy and time, the optimization may only be addressing a part of the total design. For example, the optimization problem may represent a sub-circuit of a pump controller circuit. Thus, additional unmodeled features may have to be handled in post-processing. The reasons for only looking at a partial problem are numerous. Some points include: i) No sufficiently accurate model exists, e.g., because of electro-magnetic noise in relation to circuits, ii) impossible to build a formal model of the consequences of a design, iii) misguiding of the search by problematic pairing of a high-accurate model for some objectives with a low-accurate model for other objectives, and iv) a desire to run the development project in parallel sub-teams, e.g., design pump hydraulics and electromotor at the same time.

2.2. Physical realization of a single solution

In an industrial design process, the simulation of a solution is usually followed by a prototype testing and then finally creation of machining tools for mass production. These three steps: i) simulation, ii) prototype testing, and iii) mass production each pose some challenges that make multiobjective optimization in its traditional form less favorable.

Regarding simulation, finding a large Pareto-optimal set is not always meaningful. Two nearby solutions may differ by 0.1% in performance, but the simulator may have an expected inaccuracy in the range of 1%–5% when the simulated solution is compared with a physical test of the design. Hence, it does not make sense to report a lot of performance-wise similar solutions since prototype testing of them may prove them to be statistically equal.

In prototype testing, one problem is to make a 100% accurate representation of the simulated design. All prototyping methods have tolerances. Consequently, there will be small differences between the simulated design and a corresponding prototype. In addition to this, no testing equipment is 100% accurate. Thus, repeating test will not give the exact same result.

Finally, preparing the design for mass production also renders some challenges. In simulation and prototyping, the "design changes" are caused by inaccuracies in simulator, prototype construction, and measurement. However, realization of the design may require small changes in the design, e.g., to make the part castable with a bi-directional casting process. Furthermore, mass produced parts also are not completely identical since all production processes introduce small variations and it is very costly to approach a 0% tolerance. Furthermore, the process itself may introduce some random variation such as size of welding seams which is not modeled in the simulation. Hence, the produced parts differ from the simulated even if no deliberate modifications have been made to adapt the design to production.

2.3. Decision making among large sets of solutions

Challenges related to decision making often come as either i) an abstract or general statement from a development project such as "we would like to evaluate the pros and cons of 3–5 clearly different designs", ii) local selection among similar Pareto-optimal solutions, or iii) as the DM's difficulties in interpreting the results and stating explicit preference rules.

General statements from development projects are often seen in concept studies where it is important to investigate and compare significantly different designs instead of fine-tuning the performance of a design. Hence, the DM is more interested in a small set of clearly different designs rather than a large set covering a smooth transition.

Concerning the local selection among Pareto-optimal solutions, DMs tend to favor solutions residing in the so-called knee regions of the Pareto front if no domain-specific preferences can be determined a priori. Hence, solutions not in knee regions can be omitted from the set of returned solutions as the DM is typically not willing to accept a large decrease in one objective to gain a small increase in another.

Regarding the DM's abilities, it is often the case that such persons have a non-optimization background and in some cases even a non-technical background. Thus, the DM may not have the technical skills to make the right decision among a set of 500–1000 solutions representing a smooth transition from one design type to another. Presenting a low number of 5–10 clearly different solutions is more feasible because it allows the DM to inspect each solution in great detail and make a decision without explicitly stating preference relations.

3. Features of the ideal MODCO algorithm

The observations presented in the previous section can be condensed into a number of desirable features for the ideal MODCO algorithm:

1. **Return Pareto-optimal solutions.**
   The returned solutions should be on the true Pareto front or as close as possible to it.
2. **User-defined maximal number of returned solutions.**
   Resources such as money and time sets a firm limit on the number of solutions manageable in decision making. Any MODCO algorithm should have an upper limit on number of returned solutions.
3. **User-defined performance distinctiveness of the solutions.**
   The application often state if the goal is to perform an exploratory search, to return a set of fine-tuned similar alternative solutions, or something in between.
4. **User-defined design distinctiveness of the solutions.**
   The DM or domain expert typically knows if the goal is to find clearly different designs or if this is of minor importance in the application.
5. **User-defined accuracy of simulators.**
   The accuracy of the simulators plays a key role in the later decision making step. The MODCO algorithm should incorporate any known simulator accuracies to ensure that the differences in simulated performance also show in later post-processing. More precisely, if A dominates B in optimization, then this should hold for subsequent steps (prototyping, test, etc.) in the design process.
6. **Return solutions in knees or according to DM's preferences.**
   DMs are rarely (if ever) interested in solutions not located in knee regions of the Pareto front unless domain-specific preferences can be defined. Hence, the ideal MODCO algorithm should return solutions located in knee regions or alternatively solutions that comply with the user-defined preferences.

Feature 1 is obviously required by any multi-objective algorithm. Features 2–5 constitute general preferences that can be analyzed and defined *a priori* to the optimization. Feature 6 is motivated by observations on the decision making process in more than 30 multi-objective optimization tasks carried out at Grundfos R&T. Please note that feature 3 *does not* express the DM's

preferences regarding the objectives importance, but only *how* diverse the returned solutions should be performance-wise.

Regarding features 2–5, we define the user-variables for setting these requirements on the returned set. Table 1 lists the four variables supporting the features. Naturally, it is up to the MODCO algorithm to incorporate the parameters in an actual *approach* for implementing the features.

Table 1: The list of user-variables supporting feature 2–5.

Feat.	Description	Variable
2	No. of distinct candidates	$K_{NC} \in [1 : \infty) \subseteq \mathbb{N}$
3	Design distinctiveness	$K_{DD} \in [0.0 : 1.0] \subset \mathbb{R}$
4	Performance distinctiveness	$K_{PD} \in [0.0 : 1.0] \subset \mathbb{R}$ or $[0.0 : 1.0]^M \subset \mathbb{R}^M$
5	Simulator accuracy	$K_{SA} \in [0.0 : 1.0]^M \subset \mathbb{R}^M$

The parameters K_{DD} and K_{PD} for defining distinctiveness should be interpreted as shown in figure 2. As seen, a value of 0.0 indicates a user-preference for a low distinctiveness, which corresponds to a preference for very similar solutions. A value of 1.0 corresponds to high distinctiveness, i.e., very different solutions. An intermediate value of, e.g., 0.5 represents a desire for a medium level of distinctiveness. In contrast to methods with explicit formulation of preferences, this approach allows incorporation of rather vague statements from the DM or domain expert. For example, a domain expert may say *"For this problem, I know that many somewhat different solutions have roughly the same performance."* In MODCO, such a statement can be transformed into $K_{PD} = 0.0$ meaning "roughly same performance" and $K_{DD} = 1.0$ or perhaps $K_{DD} = 0.5$ representing a desire for highly or somewhat different solutions. To allow refined control over performance distinctiveness, the K_{PD} parameter may also be stated as a vector. In this case, the DM may specify the level of distinctiveness for each objective.

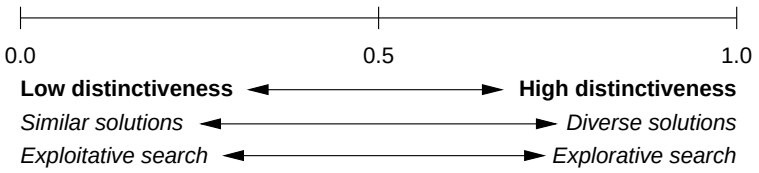


Figure 2: Distinctiveness range for parameters K_{DD} and K_{PD}.

Regarding simulator accuracy, the vector K_{SA} contains a value for each objective where 1.0 corresponds to a 100% accurate simulator and 0.0 means the returned result is no better than a random value.

3.1. Three goals of MODCO algorithms

The previous sections containing motivation and MODCO features form a basis for reformulating the three goals of traditional MO algorithms to goals for MODCO algorithms. The three goals of traditional MO algorithms are:

1. **Closeness** – the result set should be on the true Pareto front (or as close as possible).
2. **Distribrution** – the returned solutions should be evently distributed on the front.
3. **Spread** – the returned solutions should cover the entire front.

In MODCO, we suggest the following reformulation:

1. **Closeness** – the distinct candidates should be on the true Pareto front (or as close as possible).
2. **Global distinctiveness** – the candidate set should have a user-defined maximal size K_{NC} and contain clearly distinct solutions found in accordance with K_{DD}, K_{PD}, and K_{SA}.
3. **Local multi-objective optimality** – each candidate should be of primary interest to the DM, e.g., in a knee region if no explicit preferences are given.

Goal 1 of MODCO algorithms is to find solutions on the true Pareto front. This goal represent feature 1 and is identical to goal 1 in traditional MO.

Goal 2 represents the DM's desire for a set of clearly distinct solutions with a user-defined maximal number of solutions in the set. The *a priori* analysis of feature 2–5 can be summarized in one goal stating how many and how different the returned solutions should be.

Goal 3 expresses the observation mentioned in Section 2.3 regarding the DM's preference for solutions located in knee regions of the Pareto front or alternatively in accordance with the DM's explicit preferences. Thus, goal 3 represents feature 6 of the ideal MODCO algorithm.

4. Survey of MODCO-related algorithms

The MODCO approach was first studied in our earlier paper [9]. The paper contains a brief introduction to the approach, the Cluster-Forming Differential Evolution (CFDE), and a detailed study on performance and effects of changing the MODCO parameters K_{PD} and K_{DD}. The authors have also applied MODCO algorithms to three pump design problems [10].

The six features of the ideal MODCO algorithm have received some attention by other researchers.

Feature 1, closeness to true Pareto front, is clearly the most investigated since it is one of the goals of MO algorithms, for surveys see [11, 12, 13].

Feature 2 about returning a limited set of distinct solutions has been investigated in the approach known as "modeling to generate alternatives (MGA)" suggested by Brill [4] and later applied in several studies [5, 6, 7, 8]. In short, the main purpose in MGA is to present a set of maximally different solutions to allow the DM to evaluate non-modeled objectives and implicit preferences and decision rules. The idea in MGA is to first find a starting solution, then define a maximal allowed drop in fitness(es), and search one-by-one for other solutions that are maximally different in search space from previously found solutions while fulfilling the constraint of having the fitness(es) above the defined relaxation target. Although applicable for generating alternatives, the approach excludes the prime paradigm of MO – to find trade-offs wrt. the objectives. In the MGA approach, a solution just has to lie in the hypercube spanned by the relaxation thresholds. Thus, finding the maximally different solutions to accommodate the *unmodeled* objectives may result in suboptimal solutions wrt. the *modeled* objectives.

Feature 2 has also been investigated in studies incorporating the DM's preferences into the algorithm, for a survey see [3]. To the authors' knowledge, no paper suggests to integrate *general* preferences into the multi-objective algorithm with the goal of returning a user-defined maximal number of candidate solutions. However, several authors suggest to modify a traditional MO algorithm with various kinds of subpopulation schemes to increase the search performance or parallelize the MOEA, for example [14, 15]. Regarding on-the-fly clustering of solutions, Koch and Zell suggest the MOCS algorithm [16]. MOCS clusters the solutions in objective space with the aim of diversifying the search. Feature 2 is also relevant in relation to the early *interactive* evolutionary algorithms, where the evaluation and selection of solutions are decided by human interaction, and thus subject to limited re-

sources. Sims [17] suggested artificial evolution of computer graphics, and in his study it is argued that it is infeasible to select among a large number of objects in connection to a selection interface. It is argued that only a small number of objects, up to 12–16 can be visually overviewed simultaneously. This is relevant, as it corresponds exactly to the a posteriori selection of promising candidates of MOEA result sets.

Features 3 and 4 regarding setting a user-defined performance and design distinctiveness is to the authors knowledge also not investigated. Generally, traditional MO research focuses at obtaining an even population distribution, but not distinctiveness among a small set of candidates.

Feature 5 on simulator accuracy has, to some extent, been investigated in a related form known as robust design optimization. It should be noted that this research primarily deals with the manufacturing step of the design process mentioned in Section 2.2, i.e., limiting the problems originating from the tolerances in mass production. Several papers have been published on robust design and some methods incorporate the link back to the design variables, e.g., Li's work on robust optimization [18].

Feature 6 about finding solutions in knee regions has been investigated by a couple of authors in the MO research community. The most interesting study is presented by Branke et al. [19] who define two metrics (angle and a utility function) for locating solutions in knee regions. Another study is published by Rachmawati and Srinivasan [20] who suggest using a weighted sum similar to the utility function of Branke et al. [19]. Regarding incorporation of user-preferences, numerous studies have been performed, see [3].

5. The General Cluster-Forming Differential Evolution algorithm

In a previous study, we introduced and tested an algorithm called the Cluster-Forming Differential Evolution (CFDE) on a number of widely used artificial benchmark problems with two or three objectives [9]. In the earlier study, we validated the soundness of the MODCO approach and showed its performance potential and the MODCO parameters ability to control the return set. The present study introduces the General Cluster-Forming Differential Evolution (GCFDE) algorithm as an extension of CFDE. Both algorithms evolve K_{NC} subpopulations each defining a centroid in objective space and a another in design space.

The search is based on a primary selection criterion (PSC) and a secondary selection criterion (SSC), which together define a total ordering of

individuals. The primary fitness is based on discrete Pareto-ranking, while the secondary fitness is applied according to distinctiveness. The GCFDE algorithm variants are named GCFDE/PSC/SSC.

In this paper, we demonstrate the performance of GCFDE on multi- and many-objective constraint problems from mechanical and electrical engineering. To address the challenges of the investigated real-world constrained many-objective problems, GCFDE differs from CFDE on a few points. All calculations are now carried out in normalized objective and design space to ensure interval-independence. Furthermore:

1. An alternative centroid definition is introduced.
2. Primary selection now incorporates constraint handling.
3. Secondary selection now also handles design distinctiveness.

Algorithm 1 lists the pseudocode of the GCFDE algorithm, and nomenclature is given in Table 2. In pseudocode, $minObjDist(CO_i)$ denotes the function returning the minimum Euclidean distance from centroid CO_i to the nearest other centroid in normalized objective space, and $minDesDist(CD_i)$ denotes the corresponding function for normalized design space.

Table 2: Nomenclature for GCFDE.

Symbol	**Meaning**	**Symbol**	**Meaning**
P	Population	P_i	Subpopulation i
CO_i	Objective space centroid of P_i	CD_i	Design space centroid of P_i
$f(x)$	Objective vector of x	$d(x)$	Design vector of x
x	Individual	$x_{i,j}$	x at the j'th index in P_i

5.1. Centroid definition

To enhance convergence in a many-objective setting, the non-dominated individual with the best secondary fitness in P_i now defines the centroids both wrt. objectives and design parameters. Thus, when solving many-objective problems, we use the single individual with the highest secondary fitness in P_i to define centroids CO_i and CD_i, by setting $CO_i = f(x_{i,1})$ and $CD_i = d(x_{i,1})$ after truncation in the main loop. For multi-objective problems, we use the average placement of all N/K_{NC} individuals in each P_i as objective and design centroids, see [9].

Algorithm 1 General Cluster-Forming Differential Evolution (GCFDE).

Require: Population size N, K_{NC}, K_{PD}, K_{DD}
Ensure: K_{NC} distinct, feasible, non-dominated individuals.

```
Initialize K_NC subpopulations with N/K_NC random individuals.
Assign to all individuals their rank as primary fitness.
Assign to all individuals SF(x) given by preference based function.
while Halting criterion has not been met do
    Calculate all subpopulation centroids CO_i, CD_i
    Perform global DE/rand/1/bin mating with replacement
    - store incomparable offspring.
    Migrate incomparable offspring to nearest subpopulation
    wrt. Euclidean distance to objective space centroids.
    Assign to all individuals their rank as primary fitness.
    for All P_i ∈ P do
        if minObjDist(CO_i) < K_PD/K_NC then
            ∀ x_{i,j} ∈ P_i assign SF(x) according to Equation 1.
        else if minDesDist(CD_i) < K_DD/K_NC then
            ∀ x_{i,j} ∈ P_i assign SF(x) according to Equation 2.
        else
            ∀ x_{i,j} ∈ P_i assign SF(x) given by preference based function.
        end if
    end for
    Truncate subpopulations to a size of N/K_NC by sorting wrt. rank first,
    then secondary fitness.
end while
Return K_NC distinct solutions, by making a final sorting of each P_i wrt.
rank, then preference based secondary fitness, and returning x_{i,1} for i =
1...K_NC.
```

5.2. Primary selection criterion

The primary selection criterion (PSC) assigns a rank to the individuals and it thereby determines the individuals to place in the highest ranked front, i.e., those individuals that are selected wrt. the SSC.

In this paper, we use the global constraint-domination **(GCD)** relation from GDE3 [21], which states that an individual x constraint-dominates individual y iff:

- x is feasible and y is not OR
- x and y are infeasible and x dominates y in constraint space OR
- x and y are feasible and x dominates y in objective space.

5.3. Secondary selection criterion

The secondary selection criterion (SSC) determines which individuals are chosen from the highest ranked front to be included in the next generation. In GCFDE, the SSC first ensures performance distinctiveness and then design distinctiveness. However, if both performance and design distinctiveness is achieved, GCFDE performs domain-specific preference-based search. This allows concurrent application of both generalized and domain-specific preferences with MODCO parameters controlling the balance between the two. It is important to realize that the preference-based search may be defined by **any** domain-specific preference criterion without violating the desire for returning distinct candidates.

As seen in Algorithm 1, the secondary fitness assignment may differ from subpopulation to subpopulation. Thus, subpopulations that are partly overlapping in objective space select next generation based on performance distinctiveness, while subpopulations not violating neither performance nor design distinctiveness perform preference-based search.

GCFDE uses the secondary fitness assignment to fulfill distinctiveness requirements using the centroid distance measure introduced in [9]:

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

That is, we assign the minimal Euclidean distance in normalized objective space to another centroid to each individual in P_i. As this measure is to be maximized, individuals close to another subpopulation centroid will be penalized thereby guiding subpopulations away from each other if they are

too close wrt. K_{PD}. This also enhances clustering as individuals far away from their own centroid are more likely to be penalized.

In case performance distinctiveness is achieved but design distinctiveness is not, the procedure is performed in design space by assigning secondary fitness to each $x \in P_i$ according to Equation 2. Along with ensuring design distinctiveness, K_{DD} also controls to which extent the DM wishes a 1:1 correspondence between result set design and performance by guiding individuals toward clusters in both objective and design space if both K_{PD} and K_{DD} are high.

$$SF(x) = min(\{dist(d(x), CD_j), j = 1..K_{NC}, j \neq i\}) \quad (2)$$

Finally, GCFDE performs preference-based search if both performance and design distinctiveness wrt. K_{PD} and K_{DD} are achieved. In this paper, we experiment with two domain-specific preference functions. The first one is the **weighted sum (WS)** function that aggregates multiple objectives into one fitness by summing the weighted contribution of each objective using a weight vector supplied by the DM. The **WS** function guides the search toward the areas of objective space which are the most biased wrt. the weight vector supplied by the DM. Hence, the **WS** function represents explicit preferences set by the DM.

Another preference-based function is the **knee utility (KNEE)** function proposed by Branke et al. [19]. The KNEE function is designed to discover knee regions by calculating an average fitness value for a large number of randomly sampled weight vectors, 100 vectors in our experiments. The individual is more likely to reside in a knee-region if the average fitness is good. The knee utility function is generally applicable and it guides the search toward regions with good trade-offs between objectives. The KNEE function is a good choice in case the DM has no explicit preferences on the objectives.

6. Experiments and results

In general, the experimental test of an MODCO algorithm should assess its ability to fulfill the three MODCO goals 1) closeness, 2) global distinctiveness, and 3) local multi-objective optimality.

In our introducing article [9], the CFDE algorithm (GCFDE's predecessor) was tested on a large suite of standard benchmark problems, including ZDT [1], DTLZ [22] and knee problems [19]. On these problems, CFDE showed superior convergence performance compared with variants of

DEMO [23]. Furthermore, we systematically studied the effect of changing the MODCO parameters and concluded that they are able to control the result set with respect to both distinctiveness of returned solutions and also location of solutions in knee regions. Thus, all three MODCO goals were satisfactory fulfilled on these artifcial benchmark problems.

In this paper, we compare GCFDE with a number of other algorithms on a set of five well-known benchmark problems from mechanical engineering and on a many-objective real world problem from electrical engineering. This problem models a part of the control circuit for the Grundfos Alpha Pro pump, which is a small circulation pump for heating in private houses. This circuit is also used in the Alpha2 pump, see more at `www.grundfos.com/alpha2`.

6.1. Methods for performance assessment

The assessment of MODCO algorithms with respect to the three MODCO goals introduces a number of issues. First, the MODCO approach is just recently introduced and our previous algorithm CFDE is not able to handle constraints or non-normalized objective spaces, so direct comparison between MODCO algorithms is presently not possible. Second, the comparison of MODCO algorithms to traditional MO algorithms requires fair methods for comparing a low number of distinct candidates from the MODCO algorithms with a large number of solutions returned by the traditional MO algorithms.

6.1.1. Closeness goal comparison

The closeness goal of the MODCO approach is shared with the traditional MO approach. Thus, a subset of the previously published comparison methods can also be used in a MODCO context. In short, performance can be assessed using indicators as well as the standard dominance relation. Indicators are used to allow comparison between distinct candidates and their most similar counterpart from the traditional MO algorithms. In contrast, the dominance relation is used to determine to what extent the distinct candidates are dominated by *any* solution found in *all* runs of the traditional algorithms.

The indicator-based comparison of GCFDE's distinct candidates with solutions from the traditional MO algorithm is performed by calculating the Eucledian distance to the nearest solution on the approximation of the true Pareto front, which is formed using all solutions from all runs for all algorithms. To ensure a fair comparison, the distance-to-front indicator (D2F)

is calculated using the same number of solutions ($K_{NC} \cdot runs$). The D2F indicator is calculated as follows.

1. For each distinct candidate D in run i, find the solution X on the Pareto front closest to the distinct candidate D.
2. Find the solution T in run i of the traditional algorithm closest to X.
3. Add $dist(X, D)$ to the indicator for GCFDE and add $dist(X, T)$ to the indicator for the traditional algorithm.

This approach ensures a fair performance comparison in the regions near the found distinct candidates. A calculation using the entire population of the traditional algorithm would distort the comparison with the MODCO algorithm as the majority of the found solutions would cover other parts of the front. Consequently, the indicator would only allow comparisons between the traditional algorithms, but not provide any insight on performance between GCFDE and the traditional algorithms.

In the tables, each comparison between GCFDE and the compared algorithm lists the average distance to front (D2F), the result of a Mann-Whitney rank sum test (Reject/Accept H_0), the effect size (Cohen's d), the effect classification (S=small, M=medium, L=Large), the difference in rank sum (ΔR) of the two methods, and finally the percentage of GCFDE candidates dominated (Dom) by any solution in all runs of the compared algorithm. Furthermore, we determine the percentage of non-dominated distinct candidates when compared to any solution found by any traditional MO algorithm in all runs (denoted *Ndom* in the tables).

6.2. Mechanical engineering problems

To investigate GCFDE on a set of constrained multi-objective problems, we use the Two Member Truss Design (TMTD), the Gear Train Design (GTD), the Multiple Disk Clutch Design (MDCD), the Spring Design (SD) and the Welded Beam Design (WBD) problem from [24]. On these problems, we focused on the closeness goal and the ability of controlling performance distinctiveness through K_{PD}, as no local multiobjective optimality criteria exists for these artificial benchmark problems.

For each test problem, we performed 20 runs with a population size of $N = 100$ and DE parameters $F = 0.5$ and $CF = 0.3$. On all problems except the GTD problem, we have performed only 100 generations of both algorithms, whereas for the GTD problem 200 generations were performed

to ensure convergence. For the GCFDE algorithm, we have used $K_{NC} = 5$, $K_{PD} = 0.5$ and $K_{DD} = 0.0$ on all runs.

Table 3: UPDATE ALL NUMBERS!!

Algorithm	**TMTD**	**GTD**	**MDCD**	**SD**	**WBD**
GCFDE/KNEE	D2F=0.00118 NDom=34%	D2F=0.00845 NDom=74%	D2F=5.79E-5 NDom=99%	D2F=0.00379 NDom=1%	D2F=0.02513 NDom=57%
GDE3	D2F=0.00749 Reject H_0 d =1.54 (L) ΔR=-8126 Dom=36%	D2F=0.0 Accept H_0 d =- (-) ΔR=1000 Dom=26%	D2F=0.00563 Accept H_0 d =- (-) ΔR=-1314 Dom=1%	D2F=0.00479 Accept H_0 d =- (-) ΔR=-314 Dom=94%	D2F=0.05002 Reject H_0 d =0.21 (S) ΔR=-6676 Dom=33%
DEMO^{SP2}	D2F=0.00873 Reject H_0 d =1.34 (L) ΔR=-8292 Dom=47%	D2F=0.0 Accept H_0 d =- (-) ΔR=1000 Dom=26%	D2F=0.00704 Accept H_0 d =- (-) ΔR=-1314 Dom=1%	D2F=0.00583 Accept H_0 d =- (-) ΔR=-985 Dom=98%	D2F=0.05637 Reject H_0 d =0.25 (S) ΔR=-6666 Dom=36%
DEMO$^{IB}/I_{HD}$	D2F=0.22269 Reject H_0 d =1.44 (L) ΔR=-8864 Dom=21%	D2F=0.14393 Reject H_0 d =3.01 (L) ΔR=-9320 Dom=4%	D2F=0.22269 Reject H_0 d =1.44 (L) ΔR=-8864 Dom=1%	D2F=0.03509 Reject H_0 d =0.84 (L) ΔR=-3431 Dom=55%	D2F=0.22637 Reject H_0 d =1.17 (L) ΔR=-7384 Dom=31%

The results are presented in Table 3. As seen, the average distance to the Pareto front (D2F) for the GCFDE results are lower than the results from competing MOEAs in all cases except for the GTD problem. To investigate further, we conducted Mann-Whitney rank sum tests between GCFDE and the compared algorithms. The null-hypothesis is rejected in all cases for TMTD and WBD, but not for GTD, MDCD, and SD. In addition, GCFDE outcompetes DEMO$^{IB}/I_{HD}$ in all cases.

Regarding effect sizes, Cohen's d is classified as large against all other algorithms on the TMTD problem. On the WBD problem, Cohen's d is categorized as small against GDE3 and DEMO^{SP2} and large against DEMO$^{IB}/I_{HD}$.

To further illustrate the performance of GCFDE, we also note that the rank sum (ΔR) calculated in the Mann-Whitney rank sum test is negative, which favors GCFDE, in 13 out of 15 cases. Here, the maximum rank sum difference is 10000 ($100 \cdot 100$), so in most cases the difference is very significant. This tendency is also indicated by the dominance indicators, where NDom for GCFDE is overall quite good, considering that this measure tests against a Pareto front created from *all* results from competing algorithms.

In conclusion, the GCFDE algorithm is not outperformed by any of the competing algorithms, but has better performance on two problems and sim-

ilar performance on three problems. Finally it may be noted that these problems are perhaps not challenging for DE-based multi-objective algorithms as all algorithms perform quite well.

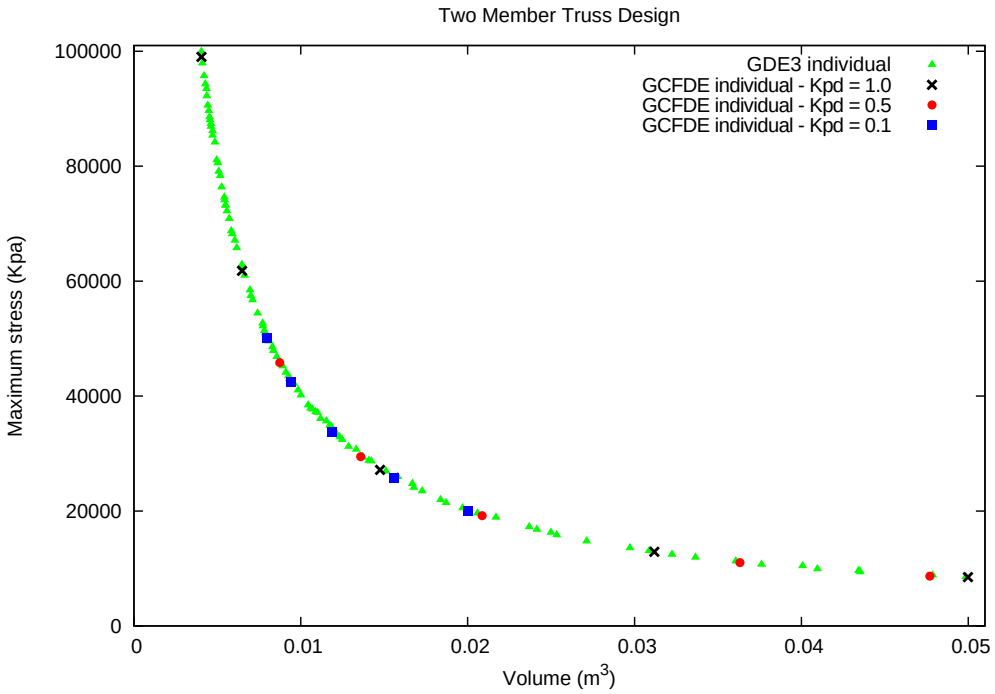


Figure 3: GDE3 and GCFDE/GCD/KNEE result sets on the TMTD problem.

To illustrate the global distinctiveness goal, the GCFDE algorithm was run with different K_{PD} settings on the TMTD problem, see Figure 3. As expected, the distance between candidates decreases when K_{PD} is lowered, while extreme solutions are found and maintained when K_{PD} is maximal, confirming our earlier observations from thorough diversity tests on artificial benchmark problems, see [9].

6.3. Circuit component sizing for the Alpha Pro pump

On the electrical engineering problem, we focused on closeness and used our DM and domain expert to set explicit preferences for goal 3. The ability to control the returned solutions by varying the MODCO parameters was not studied as each run takes approximately 48 hours and conducting a

full parametric study on this problem is therefore not feasible. Instead, we presented the found solutions to the DM and domain expert and asked for a validation of their quality. The reader is referred to our previous study [9] and Subsection 6.2 for tests on the MODCO parameters ability to control the result set.

The objective in the circuit component sizing problem is to find component values for a number of resistors and capacitors resulting in a circuit matching the desired functionality. The sub-circuit being optimized is a low-pass filter with DC rescaling functionality.

It may be possible to address this problem analytically and calculate a solution directly. However, this will most likely produce a solution having component values not available in the standard rows for resistors and capacitors. Rounding such an analytical solution to standard row values will typically decrease circuit performance. Thus, this approach is less attractive although it is widely used when manually solving component sizing problems. Instead, optimization may directly find the component values from the standard rows, see [25].

The circuit is illustrated in Figure 4. The components R1, R3, R4, R5, R6, R7, R8, R9, C1, and C6 are subject to optimization. The available resistor and capacitor values are limited to those in the Grundfos stock, i.e., components used in other Grundfos products.

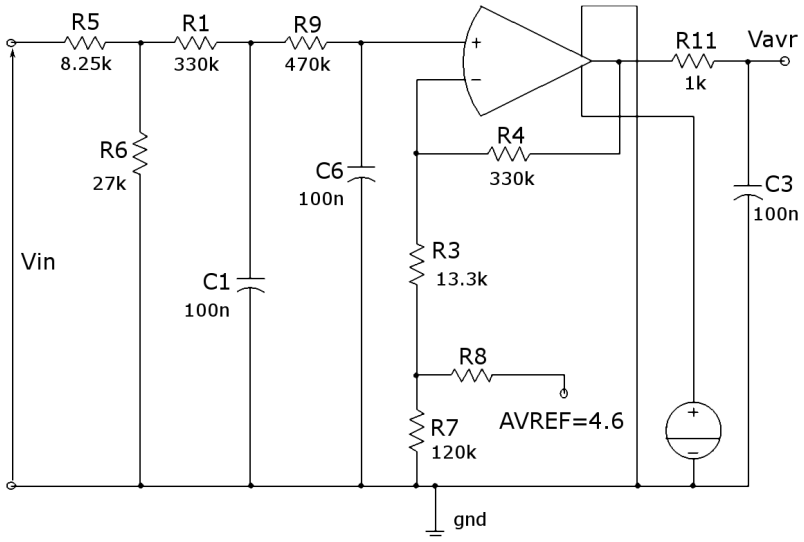


Figure 4: Circuit layout for the low-pass anti-aliasing filter.

The part of the circuit subject to optimization has both AC and DC functionality. The AC-functionality of the sub-circuit is to provide low-pass filtering of Idc=0.3 A with at least -40 dB dampening at 125 Hz. The DC-functionalities of the circuit are to attenuate V_{DC} as much as possible and to amplify simultaneously the DC-component, i.e., the average value of the signal as much as possible.

It is out of the scope of this paper to describe the problem in full details. However, a complete description with figures is available in a technical report [25]. The circuit is simulated using the Saber simulator from Synopsys. In the following, the $M(f_i)$ is the Saber magnitude function (in dB) at frequency f_i. The optimization problem has five objectives and five constraints, which are mainly for ensuring that valid component values are selected.

The first objective $F1$ captures the deviation from the desired AC functionality on certain frequencies f_i, see Table 4.

$$F1 = \sum_{f_i} \begin{cases} 20\log\left(M(f_i)\right) - v_{i,max} & 20\log\left(M(f_i)\right) > v_{i,max} \\ v_{i,min} - 20\log\left(M(f_i)\right) & 20\log\left(M(f_i)\right) < v_{i,min} \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

Table 4: Frequencies and ranges for objective $F1$.

f_i	$v_{i,min}$	$v_{i,max}$	**Comment**
0.01	8.5	10.63	Lifts AC-characteristics to desired level.
2.0	6.42	10.63	Min 2 Hz filter.
5.0	0.42	7.63	Max 5 Hz filter.

The second objective $F2$ models the dampening at 125 Hz and it is achieved by minimizing the slope around 40–50 Hz.

$$F2 = \frac{20(\log(M(50Hz)) - \log(M(40Hz)))}{50Hz - 40Hz} \tag{4}$$

Third, fourth and fifth objectives models the DC-functionality. The $F3$ objective models the gain deviation, which is extremely important to minimize as it is directly related to the tolerances of the components. A too high value on $F3$ is not acceptable as the circuit functionality will vary too much

depending on the real component values of each produced circuit. The objectives $F4$ and $F5$ model the distance to the upper and lower limit of the output DC voltage.

$$F3 = \left| 1.0 - \frac{\mathrm{R6}}{\mathrm{R5}} \cdot \frac{\mathrm{R_{TOT}}}{\mathrm{R4}} \right| \quad \mathrm{R_{TOT}} = \mathrm{R3} + \frac{\mathrm{R7} \cdot \mathrm{R8}}{\mathrm{R7} + \mathrm{R8}} \tag{5}$$

$$F4 = |4.6 - \Delta V1 - G \cdot 2.603| \quad G = \frac{1.0 + \mathrm{R4}/\mathrm{R_{TOT}}}{1.0 + \mathrm{R5}/\mathrm{R6}} \tag{6}$$

$$F5 = |0.15 + \Delta V2 - G \cdot 1.301| \tag{7}$$

The variables $\Delta V1 = 0.12$ and $\Delta V2 = 0.10$ in the above equations provide the necessary six-sigma margins for component variations.

The constraints are mainly for ensuring that valid component values are selected, see Table 5.

Table 5: Constraints.

Constraint	**Comment**
$\mathrm{R5} + \mathrm{R6} \geq 23000\Omega$	Ensures minimum values of the circuit's input impedance.
$\mathrm{R7} \geq 10000\Omega$	Limits the current consumption from VREF (Figure 4).
$\mathrm{C6} \leq 100nF$	Practical limitation on capacitor value.
$4.50 - G \cdot 2.603 \leq 0$	Ensure upper bound on six-sigma tolerance.
$0.23 - G \cdot 1.301 \geq 0$	Ensure lower bound on six-sigma tolerance.

The a priori analysis of the circuit component sizing problem resulted in the following MODCO parameter values:

- $K_{NC} = 4$ – the project manager set the max number of prototypes.
- $K_{PD} = 0.0$ – the domain expert (electrical engineer) told us that all objectives could obtain a very low value simultaneously, but this was close to impossible with traditional design methods.
- $K_{DD} = 0.5$ – the domain expert told us that similar performance could be obtained with rather different solutions as the $F3$ objective is based on ratios between component values ($R6/R5$).

In simulation, we used population size $N = 200$, $F = 0.35$, $CF = 0.2$, and the number of generations was 2000. In scaling, $[0.0, -1.0, 0.0, 0.0, 0.0]$ was used as best point and $[1.0, 0.0, 1.0, 1.0, 1.0]$ was nadir point. Furthermore, the weight vector in WS was $[1.0, 1.0, 3.0, 1.0, 1.0]$ because $F3$ is more important than the other objectives. As a single run takes approximately 48 hours, only 10 runs were performed for each algorithm variant (six variants gave around 120 CPU days in total).

6.4. Preliminary experimentation and conclusions

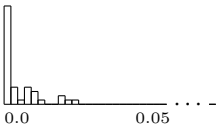
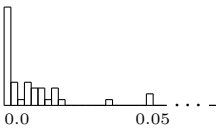
To compare the convergence capability of GCFDE and GDE3 on this problem, we use the same procedure as for the benchmark problems. As expected, the original GDE3 fails on this many-objective problem due to the absent selection pressure given only by Pareto-classification and the Crowding-distance measure of NSGA-II [1].

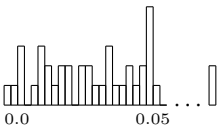
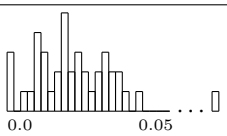
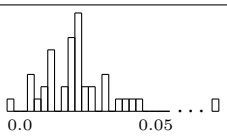
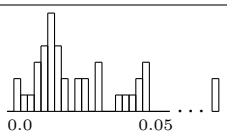
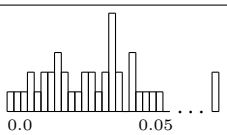
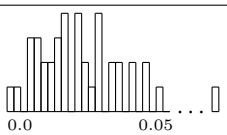
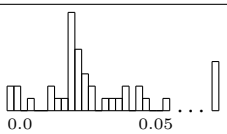
In additional preliminary runs, we tested GDE3 with two secondary selection criteria designed for many-objective optimization; the epsilon dominance indicator by Köppen and Yoshida [26] and average ranking indicator by Corne and Knowles [27]. The epsilon dominance indicator (EPS) is based on the relation of Laumanns et al. [28] and it assigns the smallest ε which would make individual x_i be dominated by all other individuals in P if added to all objectives of x_i (assuming minimization). The average ranking (AR) indicator calculates the average ranking for a solution by summing the ranks of the solution for each objective and dividing by the number of objectives. That is, a solution with an average ranking of 1 is the best on all objectives.

In these experiments, GDE3 with AR failed to produce a *single* interesting solution – even in 10 runs of the algorithm. Furthermore, the EPS showed rather inferior performance, but not direct failure. The most likely reason for the failure of AR is the $F3$ objective, in which an improvement requires five parameters to be adjusted simultaneously. The AR indicator diminish the importance of $F3$ as a large increase in $F3$ may not lead to a significant increase in overall AR score.

To enable comparison, we have used the same secondary selection criteria (WS and KNEE) in GDE3 as those used in GCFDE. Thus, the main difference between the GCFDE and GDE3 algorithms is the use of subpopulations and centroids in GCFDE. To test against previously suggested algorithms, we have also included the GDE3/EPS results in the statistical comparison.

Table 6: Results for the circuit component sizing problem.

Algorithm	**Histogram**	**D2F Indicator**
GCFDE/KNEE		D2F=0.00429 NDom=60.0%
GCFDE/WS		D2F=0.00778 NDom=65.0%

Algorithm	**Histogram**	**vs. GCFDE/KNEE**	**vs. GCFDE/WS**
GDE3/KNEE		D2F=0.02705 Reject H_0 d =1.98 (L) ΔR=-1384 Dom=7.5%	D2F=0.28205 Reject H_0 d =1.53 (L) ΔR=-1200 Dom=7.5%
GDE3/WS		D2F=0.02028 Reject H_0 d =1.66 (L) ΔR=-1234 Dom=17.5%	D2F=0.23910 Reject H_0 d =1.22 (L) ΔR=-1106 Dom=20.0%
GDE3/EPS		D2F=0.02129 Reject H_0 d =1.94 (L) ΔR=-1365 Dom=7.5%	D2F=0.02314 Reject H_0 d =1.24 (L) ΔR=-1175 Dom=15.0%
DEMO^{SP2}/KNEE		D2F=0.02416 Reject H_0 d =1.17 (L) ΔR=-1338 Dom=10.0%	D2F=0.02392 Reject H_0 d =1.18 (L) ΔR=-1067 Dom=15.0%
DEMO^{SP2}/WS		D2F=0.02549 Reject H_0 d =2.02 (L) ΔR=-1367 Dom=7.5%	D2F=0.02933 Reject H_0 d =1.44 (L) ΔR=-1238 Dom=10.0%
DEMO^{SP2}/EPS		D2F=0.02230 Reject H_0 d =1.83 (L) ΔR=-1350 Dom=22.5%	D2F=0.02554 Reject H_0 d =1.43 (L) ΔR=-1250 Dom=25.0%
DEMO^{IB}/I_{HD}		D2F=0.02986 Reject H_0 d =1.28 (L) ΔR=-1374 Dom=5.0%	D2F=0.31212 Reject H_0 d =1.16 (L) ΔR=-1230 Dom=12.5%

6.5. Results

Table 6 demonstrates that the GCFDE algorithm performs much better than the traditional MOEAs on the many-objective real world problem.

First, we note that the distance-to-front is 4-5 times lower for the GCFDE algorithm than the ones of the competing algorithms. The histograms presented in Table 6 are based on the D2F indicator, and these also show clearly that the GCFDE algorithm tend to generate results much closer to the Pareto front. This can be seen from the large amount of solutions being located in the left side of the histograms (optimum) for GCFDE, whereas for the traditional MOEAs, solutions tend to be placed in the right side of the histograms, being worse - some are even outside of the presented interval, shown as the rightmost column.

Taking a closer look at the Mann-Whitney tests, we see that H_0 is rejected *in all cases*, with large effect sizes and rank sum differences between 1000 and 1384 out of a maximum of 1600 ($40 \cdot 40$), all negative, thus significantly favoring the GCFDE algorithm.

Considering the dominance relation, the dom indicator has low values for all competing algorithms. This means that it is possible in only very few cases to find *any* solution generated by the traditional algorithm, that dominates the distinct candidate generated by GCFDE. Dom values are in the range of 5 % for the DEMO$^{IB}/I_{HD}$ algorithm and 20 % for DEMO$^{SP2}/EPS$, such that a max of one fifth of GCFDE solutions can be dominated by *any* other solution generated by any other algorithm tested. The NDom indicator takes on values around 60 % - 65 %, giving the amount of GCFDE solutions dominated by no other solution generated by the competing MOEAs. This support both the dom and the D2F indicator conclusions.

7. Conclusions

In this paper, we have introduced the MODCO approach, which is motivated by a number of observations from real-world problems from the manufacturing industry. The goal in MODCO is to return a small set of clearly distinct candidate solutions to allow a feasible in-depth post-processing of the candidates. The main idea in MODCO is to refine the multi-objective search by incorporating general preferences from the DM and the domain expert's knowledge of the problem at hand. Through a few parameters, rather vague statements from the DM or domain expert are transformed into de-

sired characteristics of the returned set of solutions. In addition to this, the MODCO approach allows incorporation of simulator inaccuracies if known.

The MODCO approach may be seen as an extension or modification of the traditional MO concept. Thus, a reformulation of the three goals closeness, distribution, and spread of traditional MO is also in place. The three MODCO goals are closeness, global distinctiveness, and local multi-objective optimality. The closeness goal states a desire to find solutions on the Pareto front, which is identical to the first goal of traditional MO. The global distinctiveness goal says that the returned set of candidate solutions should comply with the DM's general preferences on number of solutions, performance distinctiveness, design distinctiveness, and known simulator accuracy. Finally, the local multiobjective optimality goal expresses the DM's preferences for finding solutions in knee regions or according to domain-specific preferences.

To demonstrate the potential of the MODCO approach, we introduced the GCFDE algorithm and applied it to a number of real-world problems in mechanical and electrical engineering. Here, we showed the algorithms superior performance and ability to return a result set in accordance with the DM's desire for distinctiveness in the returned solutions. On the challenging many-objective circuit component sizing problem, GCFDE outperformed two other many-objective approaches in preliminary runs and as well as GDE3 with the same secondary selection criteria used in GCFDE.

Overall, this study shows that the MODCO approach is a valid and promising direction for handling multi- and many-objective real-world problems. As seen in experiments, the GCFDE algorithm was indeed able to return a few distinct candidates and at the same time improve search performance significantly. The main reason for this performance improvement is most likely the focused search balanced with the ability to spread the solutions in accordance with the DM's preferences.

Acknowledgements

The authors would like to extend their grattitude to Bogdan Filipič and Jurij Šilc for inviting our BIOMA papers to submission in Applied Soft Computing. Furthermore, we would like to thank Ville Tirronen and Matthieu Weber for providing the code generating the histograms in Figure **??**. Their code is available here: `http://users.jyu.fi/~mweber/eahist/`.

References

[1] K. Deb, Multi-Objective Optimization using Evolutionary Algorithms, John Wiley & Sons, Chichester, UK, 2001. ISBN 0-471-87339-X.

[2] D. A. V. Veldhuizen, G. B. Lamont, Multiobjective Evolutionary Algorithms: Analyzing the State-of-the-Art, Evolutionary Computation, 8 (2000) 125–147.

[3] C. A. Coello Coello, Handling Preferences in Evolutionary Multiobjective Optimization: A Survey, in: Proceedings of the 2000 Congress on Evolutionary Computation (CEC-2000), 2000, pp. 30–37.

[4] E. D. Brill Jr., The use of Optimization Models in Public-Sector Planning, Management Science, 25 (1979) 413–422.

[5] E. D. Brill Jr., J. M. Flach, L. D. Hopkins, S. Ranjithan, MGA: A Decision Support System for Complex, Incompletely Defined Problems, IEEE Transactions on Systems, Man, and Cybernetics, 20 (1990) 745–757.

[6] E. D. Brill Jr., S.-Y. Chang, L. D. Hopkins, Modeling to Generate Alternatives: The HSJ Approach and an Illustration using a Problem in Land Use Planning, Management Science, 28 (1982) 221–235.

[7] D. H. Loughlin, S. R. Ranjithan, E. D. Brill Jr., J. W. Baugh Jr., Genetic Algorithm Approaches for Addressing Unmodeled Objectives in Optimization Problems, Engineering Optimization, 33 (2001) 549–569.

[8] E. Zechman, S. R. Ranjithan, An Evolutionary Algorithm to Generate Alternatives (EAGA) for Engineering Optimization Problems, Engineering Optimization, 36 (2004) 539–553.

[9] P. D. Justesen, R. K. Ursem, Multiobjective Distinct Candidates Optimization (MODCO): A Cluster-Forming Differential Evolution Algorithm, in: Proceedings of the Fifth International Conference on Evolutionary Multi-Criterion Optimization (EMO-2009), 2009, pp. 525–539.

[10] P. D. Justesen, R. K. Ursem, Region of Interest Based Differential Evolution for Many-Objective Constrained Real-World Problems, in: Proceedings of the 2010 Congress on Evolutionary Computation (CEC-2010), 2010.

[11] C. A. Coello Coello, A Comprehensive Survey of Evolutionary-Based Multiobjective Optimization Techniques, Knowledge and Information Systems. An International Journal, 1 (1999) 269–308.

[12] C. A. Coello Coello, An Updated Survey of GA-Based Multiobjective Optimization Techniques, ACM Computing Surveys, 32 (2000) 109–143.

[13] M. Reyes-Sierra, C. A. Coello Coello, Multi-Objective Particle Swarm Optimizers: A Survey of the State-of-the-Art, International Journal of Computational Intelligence Research, 2 (2006) 287–308.

[14] J. Branke, H. Schmeck, K. Deb, R. Maheshwar, Parallelizing Multi-Objective Evolutionary Algorithms: Cone Separation, in: Proceedings of the 2004 Congress on Evolutionary Computation (CEC-2004), 2004, pp. 1952–1957.

[15] A. Molyneaux, G. Leyland, D.Favrat, A New, Clustering Evolutionary Multi-Objective Optimisation Technique, in: Proceedings of the Third International Symposium on Adaptive Systems – Evolutionary Computation and Probabilistic Graphical Models, Institute of Cybernetics, Mathematics and Physics, 2001, pp. 41–47.

[16] T. E. Koch, A. Zell, MOCS: Multi-Objective Clustering Selection Evolutionary Algorithm, in: Proceedings of the 2002 Genetic and Evolutionary Computation Conference (GECCO-2002), 2002, pp. 423–430.

[17] K. Sims, Artificial Evolution for Computer Graphics ACM Computer Graphics, 25 (1991) 319–328.

[18] M. Li, Robust Optimization and Sensitivity Analysis with Multi-Objective Genetic Algorithms: Single- and Multi-Disciplinary Applications, Ph.D. thesis, University of Maryland (College Park, Md.), 2007.

[19] J. Branke, K. Deb, H. Dierolf, M. Osswald, Finding Knees in Multi-Objective Optimization, in: Parallel Problem Solving from Nature - PPSN VIII, 2004, pp. 720–729.

[20] L. Rachmawati, D. Srinivasan, A Multi-Objective Evolutionary Algorithm with Weighted-Aum Niching for Convergence on Knee Regions, in: Proceedings of the 2006 Conference on Genetic and Evolutionary Computation (GECCO-2006), 2006, pp. 749–750.

[21] S. Kukkonen, J. Lampinen, GDE3: The third Evolution Step of Generalized Differential Evolution, in: Proceedings of the 2005 Congress on Evolutionary Computation (CEC-2005), 2005, pp. 443–450.

[22] K. Deb, L. Thiele, M. Laumanns, E. Zitzler, Scalable Multi-Objective Optimization Test Problems, in: Proceedings of the 2002 Congress on Evolutionary Computation (CEC-2002), 2002, pp. 825–830.

[23] T. Robič, B. Filipič, DEMO: Differential Evolution for Multiobjective Optimization, in: Proceedings of the Third International Conference on Evolutionary Multi-Criterion Optimization (EMO-2005), 2005, pp. 520–533.

[24] K. Deb, A. Srinivasan, Innovization: Innovative Design Principles Through Optimization, Technical Report 2005007, Kanpur Genetic Algorithms Laboratory (KanGAL), 2005.

[25] R. K. Ursem, A Many-objective Circuit Design Problem: The Anti-Aliasing Filter, Technical Report 2009-01, Grundfos R&T, 2009, `www.daimi.au.dk/~ursem/publications/TR-09-01.pdf`.

[26] M. Köppen, K. Yoshida, Substitute Distance Assignments in NSGA-II for Handling Many-objective Optimization Problems, in: Proceedings of the Fourth International Conference on Evolutionary Multi-Criterion Optimization (EMO-2007), 2007, pp. 727–741.

[27] D. W. Corne, J. D. Knowles, Techniques for Highly Multiobjective Optimisation: Some Nondominated Points are Better than Others, in: Proceedings of the 2007 Conference on Genetic and Evolutionary Computation (GECCO-2007), 2007, pp. 773–780.

[28] M. Laumanns, L. Thiele, K. Deb, E. Zitzler, Combining Convergence and Diversity in Evolutionary Multi-objective Optimization, Evolutionary Computation, 10 (2002) 263–282.