2015 IEEE Symposium Series on Computational Intelligence

# Concept-based Evolutionary Multi-Criteria Exploration of Design Spaces Under Run-time Limitation

Alon Snir, Barak Samina, Amiram Moshaiov
School of Mechanical Engineering
Tel-Aviv University
Tel-Aviv, Israel

***Abstract*— The set-based concept approach has been suggested as a means to simultaneously explore different design spaces at both the conceptual and the particular design levels. The type of exploration problem, which is dealt with here, aims to reveal the approximated concepts' fronts, within a Pareto relaxation zone. Based on a resolution-relaxation approach, a unique concept-based evolutionary algorithm has recently been suggested to tackle this problem. Here, we suggest a modification to that algorithm, which aims to provide an adaptive mechanism to distribute the computational resources among the evolved concepts under a given run-time limitation.**

## I. INTRODUCTION

The main motivation for the proposed search approach is not optimization but obtaining some general knowledge about the design space (as detailed in [1]). The work presented here is an extension of the study in [1]. For the sake of completeness, the primary ideas from [1] are described here as deemed necessary.

Our approach to design space exploration involves pre-defined design concepts that are used to explore the design space both at the level of the concepts and at the level of associated particular designs. This is achieved by the set-based concept approach. In this approach, a design concept (in short – a concept) is pre-defined by the designers as a set of potential solution alternatives, which possess some common features [2]. Such a representation has been termed Set-Based Concept (SBC). In contrast to the traditional way of evaluating concepts, the SBC approach allows concept evaluation to be based not only on optimality considerations, but also on performance variability, which is inherent to the SBC representation [3].

Fig. 1 illustrates the SBC approach. Three concepts of aircrafts are shown. The (generally different) design spaces of the concepts are marked by ellipses of different gray levels.

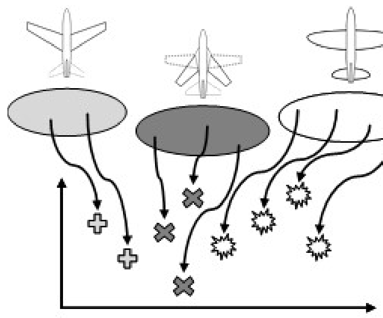
Fig. 1 Illustration of the SBC approach

The associated performance vectors of particular designs, from all concepts, are to be compared in a mutual objective space. The most studied SBC approach is known as the s-Pareto approach [4]. It involves finding which particular designs, of which concepts, are associated with the Pareto-front that is obtained by domination comparisons among all individual designs from all concepts.

In principle, existing multi-objective evolutionary algorithms can be easily used to find the front per each concept, and consequently a sorting procedure may be used to find the s-Pareto front and optimal set. However, such algorithms could also be tailored to simultaneously search all the concepts' spaces to find the same information. For example, NSGA-II, [5], served as a base for finding s-Pareto in [6], and ε-MOEA [7] was tailored for that purpose in [8]. The reader is referred to [6] and [8] for discussions on why the simultaneous search approach should be preferred over independent concept searches, and why tailoring of the algorithms is needed. Although these discussions are done in the context of an s-Pareto search, they are also valid for the current study. The reader is also referred to [6], where a detailed positioning of the SBC approach is provided with respect to related approaches.

In [3], it was suggested that concepts should not be selected by the s-Pareto approach. The alternative of a concept-based relaxed-Pareto approach was first suggested in [9], and was developed into an evolutionary algorithm in [10], using relative (dynamic) relaxation. The work in [1] follows the above studies, but extends the application of the SBC approach from

All authors are with the School of Mechanical Engineering, Tel-Aviv University, Israel (corresponding author e-mail: moshaiov@eng.tau.ac.il) concept selection to design space exploration. Reference [1] involves three main contributions including: a. a description of a general approach to design space exploration, b. a related search algorithm, and c. a method to assess the algorithm. In contrast to previous studies that suggest the SBC approach for concept selection, the work in [1], and the current study highlights its possible utilization for design space exploration.

As in [1], the current study employs SBC representations and the concept-based relaxed-Pareto approach, as a means to explore design spaces. The exploration algorithm in [1] is primarily based on two past studies on the SBC approach. The first is the study in [10], which combined the use of a relative SBC relaxation approach with NSGA-II, to produce *Cr1-NSGA-II*. The second is the study in [8], which tailored ε-MOEA to the SBC approach to produce C-ε-MOEA. However, the study in [1] substantially differs from works such as the above mainly by aiming at concept-based exploration rather than concept selection. Two types of exploration problems have been defined in [1]. Here, we concentrate on finding the concepts' fronts within a relaxation zone (first problem type), rather than just on revealing which of the concepts have solutions within that zone (second problem type).

The main contribution of the current work, as compared with [1], is the introduction of a novel method to account for run-time limitation. In general, hardware capabilities, and the averaged time for evaluating a particular design, play an important role in the total processing time that is needed to obtain meaningful results. Unfortunately, in many engineering applications the required time, to search for all the concepts' fronts, is expected to exceed the time that engineers may wait for the results. In response to the problem of the time limitation, it has been suggested in [1] to perform a simultaneous search that provides the designer with a means to reduce the computational effort by way of a pre-process assignment of a relaxation vector.

To further alleviate the aforementioned problem, we suggest a modification to the algorithm in [1]. It involves a time control mechanism, based on the difference between the allocated run-time and the on-line estimated run-time. This difference, which changes during the search process is used to dynamically influence both the required amount of computational resources and the allocation of the available resources among the concepts. It is demonstrated here that adding the suggested time control mechanism has a dramatic effect on the time required to obtain the sought information, as compared with the time it takes for the unmodified algorithm of [1].

The rest of this paper is organized as follows. In section II the foundations for the search methodology are described and in section III, the pseudo-code of the proposed search procedure, with the time control mechanism, is presented. Section IV provides a description of the benchmarking example, as well as the description and analysis of the results. Finally, section V outlines the conclusions of this paper.

## II. Problem Definition

This paper deals with the implementation of the SBC approach for design space exploration. The method has been termed, in [1], as *Concept-based Design Space Exploration* or in short C-DSE. The primary difference between using the SBC approach for such a motivation, and using it for concept selection, is in the definition of the sought information, as described in [1]. The fundamental C-DSE problem, which is dealt with here, has been described in [1]. It is restated here for the sake of completeness.

Let $n_o$ be the dimension of the objective-space $\mathbb{R}^{n_o}$ and $n_c$ be the number of concepts. Let $X_m \subseteq \mathbb{R}^{n_m}$ be the design-space of the $m$-th concept, and $f_m : X_m \to \mathbb{R}^{n_o}$ is the concept's objective-function. Let $n_m$ be the dimension of $X_m$ . Furthermore, let $s$ be any particular design and let $m_s$ and $x_s$ represent the concept index and the design vector of $s$ , respectively. Also, $x_{s,j}$ $(j = 1,...,n_m)$ represents the $j$-th element of $x_s$, and $y_s = f_{m_s}[x_s]$ represents the performance vector of $s$, with $y_{s,i}$ $(i = 1,...,n_o)$ representing the $i$-th element of $y_s$.

Without loss of generality, the *complete C-DSE*, which is based on a Pareto-approach, is about finding all the feasible Pareto-optimal solutions and front, for each problem of the following $n_c$ independent problems:

$$\min f_m[x] \quad \text{for} \quad m = 1,...,n_c \tag{1}$$

On the other hand, the s-Pareto search problem is to find the non-dominated set of the union of solutions from all the above problems. Let $P^*$ be the set of all feasible particular designs from all concepts, then the s-Pareto set is defined as follows:

$$G^* = \left\{ s \in P^* \mid \nexists s' \in P^* : \quad y_{s'} \succ y_s \right\} \tag{2}$$

Where $y' \succ y$ stands for $y'$ dominates $y$.

Let $F^*$ be the union of all the solutions of the *complete C-DSE* problem (see eq. 1). Then the first C-DSE problem type, as described in [1], is to find the concept-based relaxed Pareto-optimal set, which is defined without loss of generality, for a minimization problem, as follows:

$$R^* = \left\{ s \in F^* \mid \exists s' \in G^* : \quad y_s \succ y_{s'} + r \right\} \tag{3}$$

where vector $r = \left(r_1,...,r_{n_o}\right) \in \mathbb{R}_+^{n_o}$ is a vector of relaxation, which is defined by the users prior to the search. It is noted that the difficulty is that the relaxation zone, which contains the sought information, is relative to the s-Pareto front, which in itself is unknown prior to the search. In [1], a dynamic relaxation approach has been used, which is based on a process by which the actual relaxation during the search process decreases towards the pre-defined relaxation. Here, as explained in subsection III.D, a different approach is used, which is hereby termed as generalized relaxation.

## III. Concept-based Evolutionary Exploration

Following [1], this section starts, in III.A, with several mathematical definitions that are used in the description of the algorithm. Next, in III.B, measures are defined concerning the utilization of the computational resources and the termination condition for the algorithm. These are followed, in III.C and III.D, by a pseudo-code description of the proposed evolutionary search method, which constitutes a modification to the technique in [1].

### *A. Definition of sets used by the algorithm*

**The Population History of the *m*-th Concept** ($P_m$), is the set of all evaluated designs from all past and present iterations, which are associated with concept *m*.

**The Non-dominated Set of the *m*-th Concept** $\left(F_m\right)$, is the current set of all non-dominated designs of $P_m$ :

$$F_m = \left\{ s \in P_m \mid \nexists s' \in P_m : \quad y_{s'} \succ y_s \right\} \quad (m=1,\ldots,n_c) \quad (4)$$

Each member *s* of $F_m$ is assigned with index vector $I_s = \left(I_0, I_1, \cdots, I_{n_o}\right) \in \mathbb{Z}^{n_o+1}$ as follows:

$$I_s = \left( \left\lfloor \sum_{i=1}^{n_o} \frac{y_{s,i}}{\varepsilon_i} \right\rfloor, \left\lfloor \frac{y_{s,1}}{\varepsilon_1} \right\rfloor, \left\lfloor \frac{y_{s,2}}{\varepsilon_2} \right\rfloor, \cdots, \left\lfloor \frac{y_{s,n_o}}{\varepsilon_{n_o}} \right\rfloor \right) \quad (5)$$

$\varepsilon_i$ is the designated tolerance in the *i*-th objective below which the performance difference is insignificant to the user. Any non-empty set of designs from $F_m$ with the same index vector *I* is hereby termed as **ε-Cell**.

Each ε-cell (*e*) is assigned with a typical performance vector as follows:

$$y_e = \left(I_1 \cdot \varepsilon_1, I_2 \cdot \varepsilon_2, \cdots, I_{n_o} \cdot \varepsilon_{n_o}\right) \quad (6)$$

**The ε-Cells Set of the *m*-th Concept** ($M_m$), contains all the ε-cells of the *m*-th concept (i.e. ε-cell for each unique index vector obtained by $F_m$ members).

Here the $M_m$ set replaces the role of the ε-approximate Pareto set used in ε-MOEA [7]. This simple and low-cost method allows efficient finding of well-distributed approximation of the Pareto fronts. It is noted that the number of ε-cells occupied by each concept and the rate of occupying new ε-cells during the search process are used to support the distribution of computational resources among the deferent concepts.

**Integrated ε-Cells Set** ( $M$ ) contains ε-cells from all concepts:

$$M = \bigcup_{m=1}^{n_c} M_m \quad (7)$$

**The Global Non-dominated Set of ε-Cells** (*G*), is the set of all non-dominated ε-cells of $M$:

$$G = \left\{ e \in M \mid \nexists e' \in M : \quad y_{e'} \succ y_e \right\} \quad (8)$$

**The Relaxed Non-dominated Set of ε-Cells** (*R*), contains G and all the ε-cells of *M*, which satisfy the proximity condition:

$$R = \left\{ e \in M \mid \exists e' \in G : \quad y_e \succ y_{e'} + r \right\} \quad (9)$$

where vector $r = \left(r_1, \ldots, r_{n_o}\right) \in \mathbb{R}_+^{n_o}$ is the vector of relaxation.

### *B. Measures of resource utilization*

The following measures of resource utilization are defined to support balancing the computational resources among the various concepts, and to provide a mean for the search termination condition.

Several measures of resource utilization are defined as follows:

- The resource utilization of the ε-cell (*e*):

$$v_e = \frac{\text{SelectionCount}[e]}{SelectionLimit} \quad (10)$$

  Where, the selection count is defined as the number of times any design *s* from the ε-cell has been selected for mutation (during the evolutionary process). The selection count is limited with a predefined value (*SelectionLimit*).

- The resource utilization of the *m*-th concept:

$$v_m = \frac{1}{\left|M_m\right|} \cdot \sum_{e \in M_m} v[e] \quad (11)$$

## C. Main Procedure

```
1   Perform Initialization Procedure (see subsection III.D.1).
2   Initialize set $A$ with all ε-cells of all concepts and empty set $B$.
3   While run time didn't exceed the time limit:
    2.1   Perform Generalized Relaxation Update Procedure (see subsection III.D.3).
    2.2   Select one ε-cell ($e$) from set $A$ according to Cell Selection Procedure (see subsection III.D.2).
    2.3   Randomly select one design $s$ from $e$.
    2.4   Perform Element Mutation (see subsection III.D.4) on $s$ to produce a new design $s'$.
    2.5   Evaluate $s'$ and update all the sets that are defined in subsection III.A.
```

## D. Sub-procedures

### 1) Initialization Procedure

Procedure steps:

1 Randomly initialize a population of designs with equally sized subpopulations for each concept.

2 Evaluate each design.

3 Initialize all the sets defined in section III.A.

4 Start the evolution process by equal distribution of resources among the concept for a predefined number of iterations.

### 2) Cell Selection Procedure

To ensure a fair selection, this procedure takes into consideration the resource utilization (see section III.B) of every concept and every selectable ε-cell (from set $A$). The proposed rule is that the selected ε-cell will be the ε-cell with the lowest resource utilization from the concept with the lowest resource utilization.

Procedure steps:

1 Select the concept $m$ with the lowest resource utilization ($v_m$). If there is more than one concept with the same lowest resource utilization, then randomly select one of these concepts.

2 From the selected concept $m$, select the ε-cell $e$ with the lowest resource utilization ($v_e$). If there is more than one ε-cell with the same lowest resource utilization, then randomly select one of these ε-cells.

### 3) Generalized Relaxation Update Procedure

In [1], a relaxation decay profile has been suggested to retain wide relaxation at early stages of the evolutionary process, and then to decay on a linear slope until the final relaxation value is achieved at an advanced stage of the evolutionary process. The profile, in [1], also ensures that, once reached, the final value is retained until the end of the evolutionary process.

A general idea for a possible (not the actual) modification to the algorithm in [1], is schematically described in Fig. 2. As shown in the figure, the size of the relaxation zone may be influenced by the *"Time Error."* Namely, as the difference between the current available (desired) run-time and the current estimated run-time becomes small, then the relaxation vector should be reduced and vice-versa. Presumably, in such a case, when an envisioned algorithm predicts that there is no sufficient time to complete the search successfully, then the algorithm decreases the relaxation, which in turns decreases the required resources.

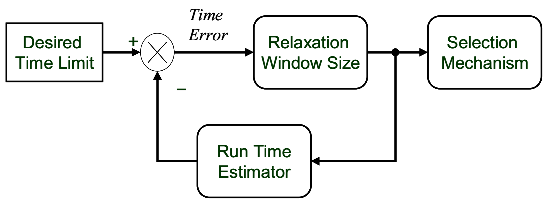

Fig. 2 Illustration of the time control method

However, here a finer approach to influence the amount of required resources, and the allocation of resources among the concepts, is suggested. It is based on the ε-cells rather than on an adaptive relaxation vector. The finer approach is hereby termed as a generalized relaxation.

Given any ε-cell $e \in M$ let $T[e]$ be the estimated needed time according to the difference between the selection-limit and its selection-count. Also let $T$ be the total estimated time of all the ε-cells in $A$ (current estimated run-time) and $T^*$ be current available (desired) run-time.

Procedure steps if $T > T^*$ (under time pressure):

1 Each ε-cell in $A$ is assigned with a scalar value derived by a non-domination ranking based on its performance vector and its resource utilization.

Let $e'$ be the worst ranked ε-cell in $A$.

2 While $T > T^*$ and $e' \notin R$ (where $R$ is the relaxed non-dominated set of ε-cells defined in III.A), shift $e'$ to $B$.

Procedure steps if $T < T^*$:

1 Each ε-cell in $B$ is assigned with a scalar value derived by a non-domination ranking based on its performance vector and its resource utilization.

Let $e'$ be the best ranked ε-cell in $B$.

2 While $T < T^*$ and $B \neq \varnothing$ and $T[e'] \leq T^* - T$, shift $e'$ to $A$.

*4) Element Mutation*

This operator produces a new offspring design $s'$ based on a parent design $s$. $x_{s'}$ will be identical to $x_s$ except of one design parameter, which is randomly selected to be mutated.

We assume the design space of each concept is bounded and divided into hyper-rectangles, which are defined according to a desired resolution. Let $x_{m,j}^{\min}$, $x_{m,j}^{\max}$ and $N_{m,j}$ be the lower limit, the upper limit and the number of resolution intervals for the *j*-th component of the design vector of the *m*-th concept, respectively. The mutation is performed as follows:

1 Initialized $x_{s'}$ to be identical to $x_s$.

2 Randomly select one design parameter $j \in 1,\dots,n_m$ and a direction indicator $t \in \{+1,-1\}$.

3 Calculate the number of resolution intervals as available for the mutation step for the selected parameter and direction as follows:

$$\Delta = \left[ \begin{array}{l|l} \dfrac{x_{s,j} - x_{m,j}^{\min}}{x_{m,j}^{\max} - x_{m,j}^{\min}} \cdot N_{m,j} & t = -1 \\ \dfrac{x_{m,j}^{\max} - x_{s,j}}{x_{m,j}^{\max} - x_{m,j}^{\min}} \cdot N_{m,j} & t = +1 \end{array} \right. \tag{12}$$

If $\Delta < 1$ select a different direction and recalculate $\Delta$.

4 Invoke random value drawn from the standard normal distribution Z, and calculate a normalized mutation step as follows:

$$\delta = \min\left[\Delta, 1 + |Z|\right] \tag{13}$$

5 Set the new value of the selected design parameter of $s'$ according to the mutation step:

$$x_{s',j} = x_{s,j} + t \cdot \frac{\delta}{N_{m,j}} \cdot \left(x_{m,j}^{\max} - x_{m,j}^{\min}\right) \tag{14}$$

## IV. Experiments and Results

### *A. Test Problem*

Similar to the idea that is presented in [1], also here, common test functions are transformed by translation and scaling in the objective space, as seen in 0 In contrast to the bi-objective example in [1], which involved ten-concepts, here the devised bi-objective test problem involves forty-concepts. Each concept's design-space and objective-function are based on one of the standard test-functions, which are commonly used for the development of multi-objective evolutionary algorithms (e.g., [5]). Table I provides details for forty concepts, which are based on nine test functions with different transformations. For example, FON is used to produce four different functions by four different transformations, which correspond to four different concepts. Each transformation type (scale and offset) is presented, in Table I, as a row vector with the 1st and 2nd components corresponding to the 1st and 2nd objective, respectively. Fig. 3 shows all forty fronts. In the current study, the size of each component of the relaxation vector is one. This can easily be observed as the gray area in Fig. 4, which shows a part of the previous graph in an enlarged scale. Clearly, many of the concepts' fronts are actually out of the relaxation zone.

Obtaining all the fronts as shown in Fig. 3 took approx. four hours on a PC with six cores. This is because we have incorporated a time delay per each evaluation of a design, to make the search characteristics similar to a real-life problem search, in which the evaluation time may be significant and it may defer from one concept to the other. In the example below, the delays ranged between 0.06 to 2 seconds per an evaluation of an individual solution, such that each concept had a different time-delay associated with it.

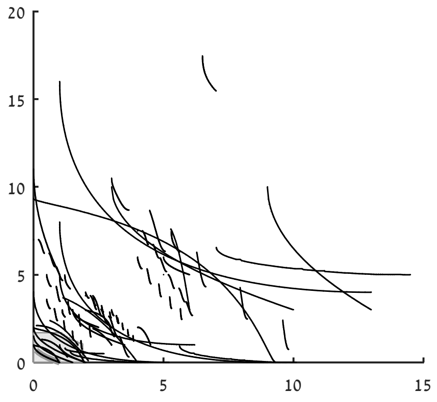

Fig. 3 The concepts' fronts

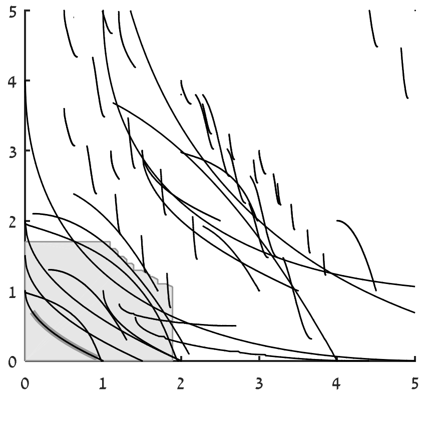

Fig. 4 Some of the concepts' fronts (enlarged scale)

TABLE I CONCEPTS' DETAILS

| **Test Fn.** | $n_m$ | **Design-Space Bounds** | **Obj. Scale** | **Obj. Offset** | $N_m$ |
|---|---|---|---|---|---|
| FON | 3 | [ −4, 4 ] | [ 1.0, 1.0 ] | [ 0.0, 0.0 ] | 40 |
| | | | [ 2.0, 2.0 ] | [ 0.0, 0.0 ] | |
| | | | [ 1.0, 1.0 ] | [ 3.0, 3.0 ] | |
| | | | [ 9.5, 9.5 ] | [ 0.0, 0.0 ] | |
| KUR | 3 | [ −5, 5 ] | [ 0.3, 0.3 ] | [ 8.3, 8.0 ] | 200 |
| | | | [ 0.2, 0.2 ] | [ 6.0, 3.8 ] | |
| | | | [ 0.2, 0.2 ] | [ 8.0, 7.5 ] | |
| | | | [ 0.2, 0.2 ] | [ 4.0, 7.0 ] | |
| | | | [ 0.2, 0.2 ] | [ 9.0, 5.0 ] | |
| POL | 2 | [ −π, π ] | [ 0.2, 0.2 ] | [ 1.0, 0.0 ] | 100 |
| | | | [ 0.1, 0.1 ] | [ 1.0, 0.5 ] | |
| | | | [ 0.5, 0.5 ] | [ 6.0, 5.0 ] | |
| | | | [ 0.3, 0.3 ] | [ 5.0, 0.0 ] | |
| SCH 1 | 1 | [ −1000, 1000 ] | [ 1.3, 1.0 ] | [ 0.0, 0.0 ] | 4x $10^4$ |
| | | | [ 1.3, 1.0 ] | [ 1.0, 1.0 ] | |
| | | | [ 2.0, 2.0 ] | [ 1.0, 0.0 ] | |
| | | | [ 3.0, 3.0 ] | [ 1.0, 4.0 ] | |
| ZDT 1 | 30 | [ 0, 1 ] | [ 1.0, 1.0 ] | [ 1.5, 2.0 ] | 11 |
| | | | [ 2.0, 2.0 ] | [ 1.5, 1.0 ] | |
| | | | [ 1.0, 1.0 ] | [ 5.0, 5.0 ] | |
| ZDT 2 | 30 | [ 0, 1 ] | [ 1.0, 1.0 ] | [ 0.3, 0.3 ] | 11 |
| | | | [ 2.0, 2.0 ] | [ 0.1, 0.1 ] | |
| | | | [ 0.5, 1.0 ] | [ 4.0, 1.0 ] | |
| ZDT 3 | 30 | [ 0, 1 ] | [ 1.6, 1.6 ] | [ 0.5, 2.0 ] | 101 |
| | | | [ 2.0, 2.0 ] | [ 0.5, 3.0 ] | |
| | | | [ 1.0, 1.0 ] | [ 3.0, 2.0 ] | |
| | | | [ 2.0, 2.0 ] | [ 4.0, 4.0 ] | |
| | | | [ 1.5, 1.0 ] | [ 2.0, 3.0 ] | |
| | | | [ 8.0, 5.5 ] | [ 3.0, 5.0 ] | |
| ZDT 4 | 10 | $x_1 \in$ [ 0 , 1 ] $x_{2,\ldots,10} \in$ [ −5 , 5 ] | [ 2.0, 2.0 ] | [ 0.0, 0.0 ] | 21 |
| | | | [ 1.0, 1.0 ] | [ 0.0, 0.0 ] | |
| | | | [ 1.5, 1.5 ] | [ 0.0, 0.0 ] | |
| | | | [ 1.0, 1.0 ] | [ 1.0, 0.0 ] | |
| | | | [ 7.0, 7.0 ] | [ 3.0, 3.0 ] | |
| | | | [ 1.0, 6.0 ] | [ 0.0, 5.0 ] | |
| | | | [ 4.0, 7.0 ] | [ 9.0, 3.0 ] | |
| ZDT 6 | 10 | [ 0, 1 ] | [ 1.0, 1.0 ] | [ 2.0, 1.0 ] | 41 |
| | | | [ 1.5, 1.5 ] | [ 0.2, 1.0 ] | |
| | | | [ 4.0, 4.0 ] | [ 0.0, 0.0 ] | |
| | | | [ 1.0, 5.0 ] | [ 5.0, 3.0 ] | |

## *B. Results*

The proposed algorithm has been tested as follows. First, the desired time-limit was set into three hours, which amounts to 3/4 of the time that it took to produce all the fronts. Fig. 5 shows the obtained fronts when the algorithm is used with *Selection-Limit* $= 40$, $r = [1,1]$, and $\varepsilon = 0.1$. Fig. 6 shows a closer look at a part of the objective space.

In both Fig. 5 and 6, the curves show the discovery rate of the fronts, based on eight independent runs. For example, when a curve is shown in black it means that the part of a front, which is represented by that curve, has been found in all eight runs. On the other hand, a green part means that only in four out of the eight runs that part of the front was discovered. It is important to note that the sought information is within the gray area of Fig. 6, namely within the relaxation zone. It is clear from Fig. 6 that most of the information, within that zone, was obtained in all runs, and all of it was obtained in at least six runs out of the conducted eight runs. Clearly, as seen in Fig. 5, the reduced computational resources influenced primarily the results far from the relaxation zone.

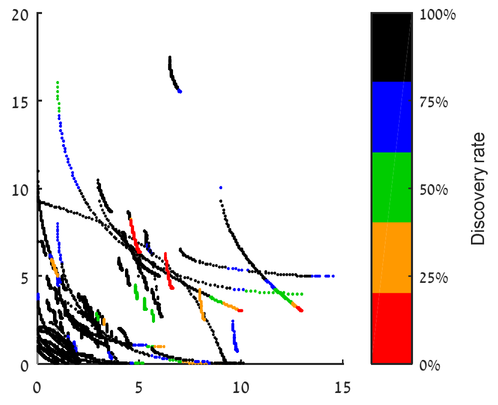

Fig. 5 Revealed concepts' fronts (time-limit = 3 hours)

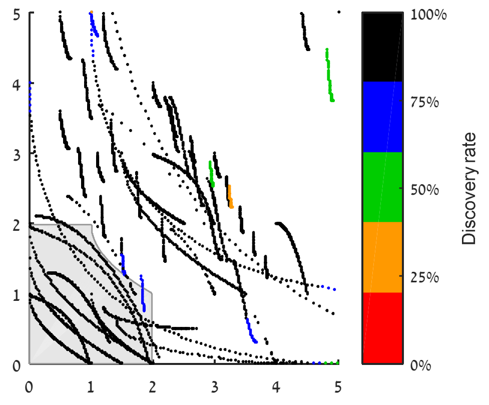

Fig. 6 Part of the revealed concepts' fronts (time-limit = 3 hours)

Next, the time limit was set into two hours, which corresponds to half of the required time to find all fronts. The results under this restriction are shown in Fig. 7 and 8. The results for a one hour restriction (1/4 of the required time) are shown in Fig. 9 and 10.

It is clear from the above series of results that the algorithm performs as it was designed for. Namely, the results in the relaxation zone, which is the zone of interest, are much less influenced by the time restriction, as compared with the results outside that zone.

## V. SUMMARY AND CONCLUSIONS

This paper employs a non-traditional set-based concept approach to simultaneously explore different design spaces as associated with design concepts. A relaxation vector is pre-defined in a mutual objective space. It serves to find a corresponding relaxation zone as related to the s-Pareto front. The exploration problem, which is dealt with here, is to find all the concepts' fronts that are within that zone. The difficulty is that the zone is relative to the s-Pareto front, which in itself is unknown prior to the search. In the current study the problem is to be solved under a restricted run-time.

Following the study in [1], the current work provides a solution to the aforementioned problem. It employs a time-control mechanism, which is shown to allocate the limited computational resources in a way that has a smaller effect on the revealed information within the relaxation zone, as compared with the obtained information outside that zone.

The above conclusion is based on the results of the given test-case. While complicated in a substantial way, this test-case must be amended with additional studies using other test-cases. It is further noted that there is no common benchmarking method for the type of problems that are presented in [1] and in the current study. Different transformations of common test functions may serve to devise alternative test-cases.

Finally, it should be noted that we have recently tested the algorithm of [1] on a real-world application composed of simultaneous exploration of 180 concepts concerning the design of aircraft propellers. That study involved a run that took a few weeks on a PC. In the near future we plan to test that real-life problem with the current algorithm under various time-restrictions. Moreover, in the long-run we plan to modify the current algorithm into an interactive one.

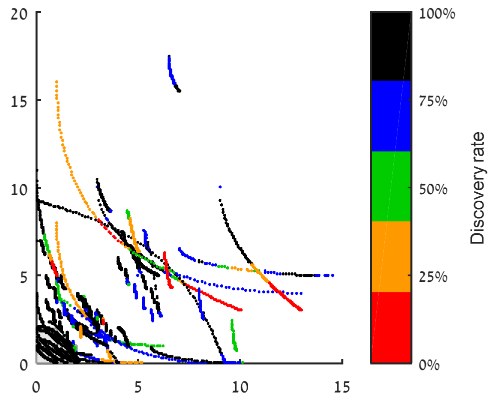

Fig. 7 Revealed concepts' fronts (time-limit = 2 hours)

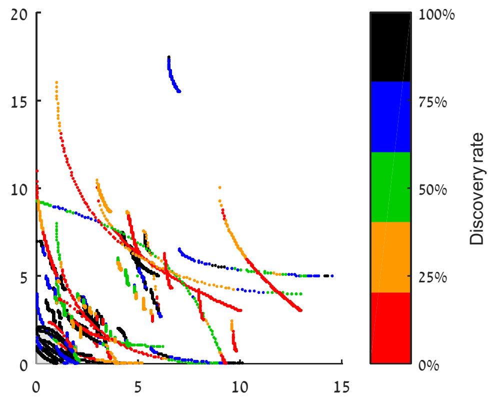

Fig. 9 Revealed concepts' fronts (time-limit = 1 hour)

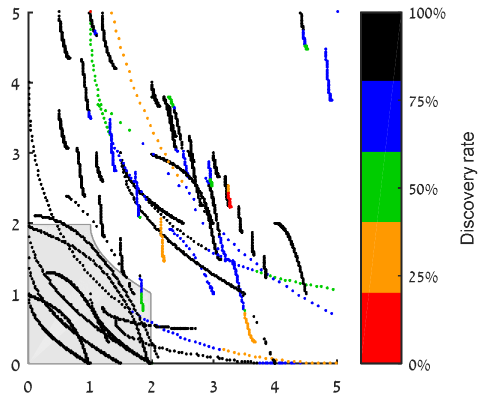

Fig. 8 Part of the revealed concepts' fronts (time-limit = 2 hours)

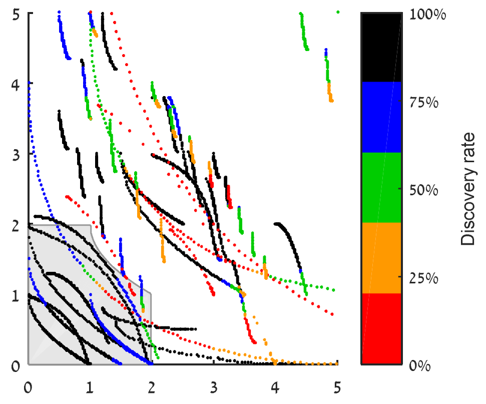

Fig. 10 Part of the revealed concepts' fronts (time-limit = 1 hour)

## REFERENCES

[1] Moshaiov, A., Snir, A. and Samina, B. Concept-based evolutionary exploration of design spaces by a resolution-relaxation-Pareto approach. To appear in the Proc. of the IEEE Congress on Evolutionary Computations, 2015.

[2] Moshaiov, A., and Avigad, G. Concept-based multi-objective problems and their solution by EC. Short paper, Proc. of the GECCO 2007 Workshop on User Centric Evolutionary Computation, 2007.

[3] Avigad, G., and Moshaiov, A. Set-based concept selection in multi-objective problems: optimality versus variability approach. J. of Eng. Design, Vol. 20, No. 3, pp. 217-242, 2009.

[4] Mattson C. A., Messac, A. Pareto frontier based concept selection under uncertainty with visualization. Opt. and Eng., 6; p. 85–115, 2005.

[5] Deb, K., Pratap, A., Agarwal, S., Meyarivan, T. A fast and elitist multiobjective genetic algorithm: NSGA-II. Evolutionary Computation, IEEE Tran., vol. 6, pp. 182-197, 2002.

[6] Avigad, G., Moshaiov, A. Simultaneous concept based evolutionary multiobjective optimization. Applied Soft Computing, Vol. 11, No. 1, pp. 193-207, 2011.

[7] Deb, K., Mohan, M., Mishra, S. Evaluating the ε-domination based multi-objective evolutionary algorithm for a quick computation of Pareto-optimal solutions. Evolutionary Computation, Vol. 16, No. 3: pp. 355-384, 2005.

[8] Moshaiov, A., and Snir, Y. Tailoring ε-MOEA to concept-based problems. Parallel Problem Solving from Nature - PPSN XII, Lecture Notes in Computer Science, LNCS 7492, pp. 122-131, 2012.

[9] Moshaiov, A., and Avigad, G. The extended concept-based multi-objective path planning and its A-life implications. Proc. of the 1st IEEE Symposium on A-life, in 2007 IEEE Symposium Series on Computational Intelligence, 2007.

[10] Denenberg, E., and Moshaiov, A. Evolutionary search of optimal concepts using a relaxed-Pareto-optimality approach. Proc. of the IEEE Congress on Evolutionary Computations, pp. 1093 - 1100, 2009.