Dynamic Distance Minimization Problems for dynamic Multi-objective Optimization

Heiner Zille, André Kottenhahn and Sanaz Mostaghim

Faculty of Computer Science, Otto von Guericke University Magdeburg, Germany
Email: {heiner.zille, sanaz.mostaghim}@ovgu.de, andre.k.science@gmail.com

***Abstract*—In this article we propose a new dynamic multi-objective optimization problem. This dynamic Distance Minimization Problem (dDMP) functions as a benchmark problem for dynamic multi-objective optimization and is based on the static versions from the literature. The dDMP introduces a useful property and challenge for dynamic multi-objective algorithms. Not only the positions of the Pareto-optimal solutions in the search space change over time, but also the complexity of the problem can be adjusted dynamically. In addition the problem is based on a simple geometric structure, which makes it useful to visualize the search behaviour of algorithms. We describe the basic principles of the problem, and introduce the possible dynamic changes and their implementation and effects of the Pareto-optimal areas. Our experiments show how a possible instance of the dynamic DMP can be defined and how different algorithms react to the dynamic changes.**

I. Introduction

In recent years, many-objective and large scale problems have been studied in the area of evolutionary multi-objective optimization (EMO). For evaluating the performance of EMO algorithms, scalable test problems have been introduced. In addition to the field of static multi-objective optimization, an interest in research lies also in dynamic environments. Various applications in the real world deal with dynamically changing systems, and consequently there has been research to solve (multi-objective) optimization problems, when the problems' properties change over time. One challenge in evaluating the solutions of such problems for large number of objectives concerns the visualization of the results.

Distance Minimization Problems (DMP) have been introduced as (static) scalable test problems which can be easily visualized in the objective space. This kind of problem contains several predefined objective points (the same as the number of objectives) in the decision space. The goal is to find the solutions in the decision space which have the minimum distances to all of these objective points.

In previous research this problem was used to visually demonstrate certain search behaviour of algorithms [1], which is a desirable property of a benchmark function to aid the design and analysis of new algorithms. However, other research also showed that changing certain parameter, like the used distance metric, can result in a drastic change in complexity and structure of the Pareto-optimal sets [2].

In this work, we propose a dynamic version of the Distance minimization problem as a benchmark for dynamic multi-objective optimization. This problem is based on the existing static DMP and aims to make most aspects of the DMP changeable over time. This new dynamic problem has the advantage of being easy to understand for algorithm designers as well as easily visualized to understand algorithms' search behaviour. Finally, unlike some other dynamic benchmark problems, by changing the distance metric, the problem's complexity changes drastically, which can be an interesting challenge to optimization algorithms.

The remainder of this paper is structured as follows. First, we will give a short overview of existing research in dynamic multi-objective benchmarks and the research about the static version of the DMP. In Section III we describe the static DMP and its properties. Section IV deals with the proposed changes and extensions to the problem and discusses the implementation and implications on the problems properties. Finally, we will show experimental data to examine the search behaviour of two dynamic algorithms exemplary on an instance of the new dynamic problem.

II. Related Work

Some of the existing static benchmarks like the ZDT-problems [3] are scalable in terms of the number of variables while others like the DTLZ [4] or WFG [5] are scalable both in terms of the number of variables and objectives. Similar to the benchmark problems for static optimization, multiple dynamic benchmark problem families have been proposed like the FDA [6], DTF [7], the dMOP-functions [8] or the T1- to T4-function [9]. Some of these are extensions of previous static versions. A detailed review of dynamic benchmark functions can be found in [10].

A basic version of the Distance Minimization Problem (DMP) was introduced in [11] and [12] and has been reformulated in other research since then. Schütze et al. first formulated the problem for arbitrary numbers of variables and objectives [13].

Different versions of the DMP, sometimes also with multiple Pareto-optimal areas, were used in [1], [2], [14], [15] and [16]. In [15], an instance of a DMP was created from a real-world map to determine optimal living positions within a city, using the Euclidean metric as an approximation of

978-1-5090-4601-0/17/$31.00

the distances within the map. In [1] Ishibuchi et al. used the DMP to visually examine the search behaviour of algorithms and applied existing algorithms such as NSGA II, SPEA2 and MOEA/D to problem instances of 2 and 4 objectives with up to 1000 variables. They found that the increase in the number of decision variables had a negative effect on the diversity of solutions, and showed that the solutions converged towards the Pareto-front with a low diversity and spreading out more along the front once it was reached. This research was extended in [16] to 6- and 8-objective instances, and showed that an increase in the number of decision variables has a large influence on solution quality compared to the increase in the number of objectives.

Most previous studies on the Distance Minimization Problem have considered Euclidean distance measurement, with the exceptions of [13], [17] and [2]. In real world applications, the assumption that distances are Euclidean might not be a realistic approximation for the actual distances. For instance in logistic applications such as in robot-driven warehousing applications, or grid-like scenarios like street networks, often other measurements are employed.

The idea to use different metrics in the distance measurement of this family of problems was initially suggested in [17]. The work in [2] provided detailed analysis of the specific properties of the problem when using Manhattan-distances. It was shown in these works that the Manhattan metric drastically changes the properties of the DMP and adds to the difficulty of the problem. Using the Manhattan metric, the DMP provides a hard to solve problem even with only two decision variables and 3 objectives for the NSGA-II [18] and the SMPSO [19] algorithms [2].

III. Problem description

In this section we describe the static Distance Minimization Problem (DMP) and its properties. We start with a description of the basic static problem as introduced in [2], and propose the extension to a dynamic problem and its implications in Section IV.

The static DMP is a scalable multi-objective optimization problem which contains a set of predefined so-called *objective-points* or *target-points* $\{\vec{Z}_1, .., \vec{Z}_m\}$ with coordinates $\vec{Z}_i = (z_{i1}, .., z_{in})^T$ in the n-dimensional decision space. The number of objective-points corresponds to the number of objectives (m). The goal of the DMP is to find a set of solutions ($\in \mathbb{R}^n$) in the decision space which have a minimum distance to all of the objective-points. The general DMP is formulated as follows:

$$\begin{aligned} min \quad & f(\vec{x}) = (f_1(\vec{x}), ..., f_m(\vec{x}))^T & \\ s.t. \quad & f_i = dist(\vec{x}, \vec{Z}_i) & \forall i = 1, .., m \\ & x_j \leq x_{max} & \forall j = 1, .., n \\ & x_j \geq x_{min} & \forall j = 1, .., n \end{aligned} \tag{1}$$

The central aspect in the DMP is the function for calculating the distance ($dist(\vec{x}, \vec{Z}_i)$) between a solution vector $\vec{x}$ and the objective-point $\vec{Z}_i$. Most of the previous research has used the Euclidean metric for measuring the distances in the decision space. This metric refers to the naturally shortest distance. It is induced by the p_2 norm $\| \vec{a} \|_2 = \sqrt{\sum_{i=1}^{n} |a_i|^2}$ and will therefore also be addressed as the p_2 metric in this paper. It gives the distance of two points as:

$$dist_2(\vec{a}, \vec{b}) := \| \vec{a} - \vec{b} \|_2 = \sqrt{\sum_{i=1}^{n} |a_i - b_i|^2} \tag{2}$$

Although this might be the most natural perception of distance, it is not the only one that occurs in applications and theory. Some other work has used the Manhattan metric (also called p_1 metric), which is induced by the p_1 norm [2], [17]. With the Manhattan metric, the distances between two points in the decision space is measured as follows:

$$dist_1(\vec{a}, \vec{b}) := \| \vec{a} - \vec{b} \|_1 = \sum_{i=1}^{n} |a_i - b_i| \tag{3}$$

In addition to those two metrics, we can generalize the DMP for any metric induced by the p-norm

$$dist(\vec{a}, \vec{b}) := \left[\sum_{i=1}^{n} |a_i - b_i|^p \right]^{\frac{1}{p}} \tag{4}$$

where p can be any real number ≥ 1. In the remainder of this work, we will focus on metrics induced by p-norms, although in theory every metric can be used. The advantage of the p-norms is, that by changing just one parameter p, the problem's Pareto-front can be changed. This is regarded a one type of dynamic change in the proposed dynamic DMP and will be examined later in Subsection IV-B.

A. Pareto-fronts of the DMP

It has been shown in the literature [1], [2] that in the case of the Euclidean Distance measurement ($p = 2$), the whole Pareto-optimal solution set P (in the decision space) is defined by the convex hull of the objective-points in the search space. For instance, suppose $m = 3$ objectives and $n = 2$ decision variables are used with the Euclidean metric, we obtain the Pareto-optimal solutions P as shown in Fig. 1.

Regarding the Manhattan metric based DMP ($p = 1$), it was shown in [2], that the Pareto-optimal front can not be calculated in the same way any more. The Manhattan-metric DMP shows degenerated Pareto-fronts as well as a domination structure that makes it very difficult for Pareto-dominance based algorithms to approach the true Pareto-front.

To obtain the Pareto-optimal solution sets of a Manhattan-metric DMP, a more complicated procedure is used to obtain the fronts analytically for certain 2-dimensional instances of the problem. For detailed information on how these fronts can be obtained, please refer to [2]. In Fig. 2, the Pareto-optimal sets in the decision space for 2-dimensional problem with $m = 3$ objectives is shown. For the three objective-points Z_1, Z_2, Z_3 the Euclidean (a) and the Manhattan metric (b) is used as the

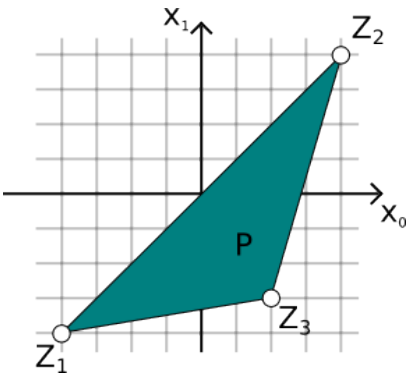
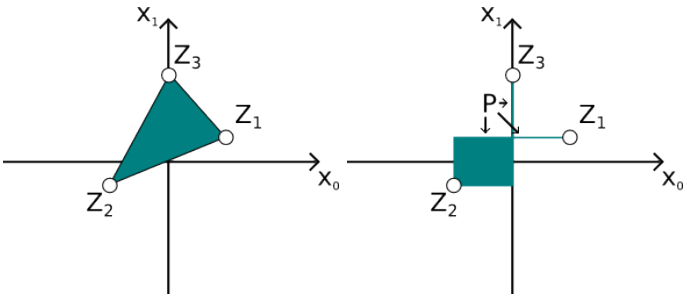


Fig. 1. Decision space of the Euclidean DMP for $m = 3$ objectives and $n = 2$ decision variables. The convex triangle (P) is Pareto-optimal.



(a) Euclidean metric (b) Manhattan metric

Fig. 2. Pareto-optimal solution in a 2-dimensional decision space with $m = 3$ objectives.

distance measure and the Pareto-optimal solutions are shown in the figures.

An essential insight gained from [2] is also, that the complexity of the DMP increases when the Manhattan metric is used instead of the Euclidean one. In the mentioned study, the NSGA-II, MOEA/D and SMPSO algorithms were not able to approximate the Pareto-optimal front of even a 2-dimensional problem when Manhattan distances were used. On the other hand, in multiple other studies [1], [14], [15] and [16] it can be seen that the respective Euclidean problem is easy to solve for current algorithms.

IV. The Dynamic Problem

In this section we will go into detail on the different types of changes and their implications for the DMP. In the following, we will refer to the dynamic DMP as *dDMP*. Since a dynamic optimization problem changes its properties over time, the DMP which was shown in Equation 1 is extended by a time-component t as follows.

$$\begin{aligned} min \quad & f(\vec{x},t) = (f_1(\vec{x},t),...,f_{m(t)}(\vec{x},t))^T & \\ s.t. \quad & f_i(\vec{x},t) = dist(\vec{x},\vec{Z}_i(t),t) & \forall i = 1,..,m(t) \\ & x_j \leq x_{max} & \forall j = 1,..,n \\ & x_j \geq x_{min} & \forall j = 1,..,n \end{aligned} \tag{5}$$

The parameter t represents the time-component in real-world applications. In the context of this work we will make t dependent on the number of already used function evaluations for optimizing the problem. This means, which each time the problem is evaluated by an optimizer, a function will map this number of evaluations to a time-parameter t that defines certain (periodic) changes of the DMP. In this work, we propose the dDMP with four different time-dependent changes to the DMP:

1) Change the position of objectives by
   a) Rotation
   b) Translation
2) Change the distance metric
3) Change the number of decision variables
4) Change the number of objectives

In the following, we will go into detail on each of these cases and describe the change in the problem's difficulty and other properties.

A. Change of Location

The most natural change that we can make based on the geometric properties of the problem is the change of the positions of the objective-points in the decision space. Since they have a fixed location in the n-dimensional search space for the static DMP, we can use translation or rotation of all points Z_i, $i = 1..m$ to move the objective-points, and thereby the Pareto-optimal set P to different locations or rotate it. As a result, solutions which were Pareto-optimal before a change (at time $t-1$) might not be optimal afterwards (at time t) any more and vice versa. This behaviour in a 2-dimensional space is visualized in Fig. 5. In the following we describe these changes formally.

1) Rotation: Rotation can be achieved by applying a n-dimensional rotation matrix $\hat{R}$ to the original objective-points at time $t = 0$ to achieve the locations at timestep t. For a 2-dimensional search space this can be expressed as

$$Z_i(t) := \hat{R} \cdot Z_i(0) \tag{6}$$

with

$$\hat{R} := \begin{pmatrix} \cos(\alpha(t)) & -\sin(\alpha(t)) \\ \sin(\alpha(t)) & \cos(\alpha(t)) \end{pmatrix} \tag{7}$$

In theory, each Z_i can be multiplied with its own rotation matrix $\hat{R}_i$. However, for simplicity, we will assume all Z_i are rotated with the same rotation matrix $\hat{R}$. The parameter for the rotation $\alpha(t)$ will be subject to change over time, so that each new objective-point is computed out of the original one at $t = 0$.

An alternate approach for practical reasons can also be applied in for rotation. Since large-dimensional rotation matrices might be hard to handle, it is also possible to define the original coordinates at time $t = 0$ in a polar coordinate system. In this way, the original positions of each Z_i consists of $n-1$ angles $\phi_1,..,\phi_{n-1}$ which define the orientation of the point in the space and one number r that defines the distance of it

from the origin. This has the advantage, that rotation can be applied in an easy way by making the coordinates ϕ_j directly dependent on the time t, i.e. we obtain for the Z_i:

$$Z_i := \begin{pmatrix} r \\ \phi_1(t) \\ \vdots \\ \phi_{n-1}(t) \end{pmatrix} \tag{8}$$

With this definition, we can directly influence the rotation in any n-dimensional decision space in an easy way. The only additional operation necessary in this case is the transformation back to the Cartesian coordinate system before adding the translation vector (see the following subsection).

2) Translation: The second movement of the objective-points is the translation, which can be expressed as a translation vector $v(t)$ that adds to the original coordinates of each point at $t = 0$. Similar to the rotation case, one could define one translation function $v_i(t)$ for every Z_i. However, in order to shift the whole Pareto-optimal set by a certain amount without changing its shape, we can add the same translation vector $v(t)$ to every objective point. Formally, the translation is described by:

$$Z_i(t) := Z_i(0) + v(t) \tag{9}$$

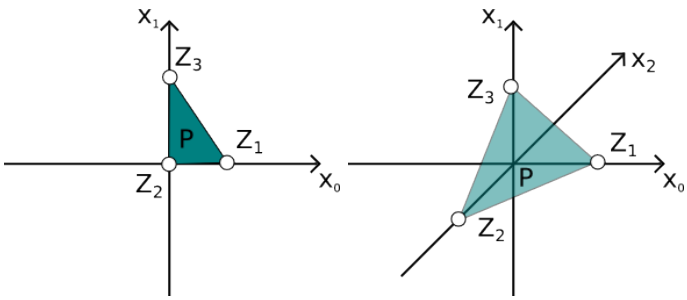
B. Change of Distance Metric

One of the most interesting things to change in the dDMP is the used metric for measuring the distance. As mentioned before, it was shown in the literature that for instance the Manhattan-metric DMP has different and more complicated Pareto-optimal sets than the same problem using the Euclidean distance. To make soft transitions between different metrics in the dDMP, we limited this work to the usage of the p-norm induced metrics as shown in Equation 4. To account for the change, we make the parameter p time-dependent, i.e. we use $p(t)$.

C. Change the Number of Variables

Another implemented change regards the number of decision variables. This means that during the optimization of the problem, the dimensionality of the search space changes. For an optimization algorithm which optimizes the 2-dimensional problem, it might suddenly be confronted with a 3-dimensional search space (see Fig. 3). This sudden increase in the dimensionality might provide a great challenge for optimization algorithms, especially when the location of the objective-points changes as well, as they are placed in a higher-dimensional space.

The challenge for the practical implementation is here, that most black-box optimizers are not meant to deal with the change in dimensionality. To implement an insertion or deletion of decision variables, the optimization algorithm has to be adapted as well to recognize the change and allocate new space for the additional variables as well as find initial values for the new variables for all existing solutions in the



(a) Two decision variables (b) Three decision variables

Fig. 3. Change of the Pareto-optimal solutions in the decision space with 3 objectives in response to an increase in the number of variables.

population. As we are aiming to provide a black-box problem, we do not expect this kind of behavior from algorithms. Therefore, we implemented this change in the following way.

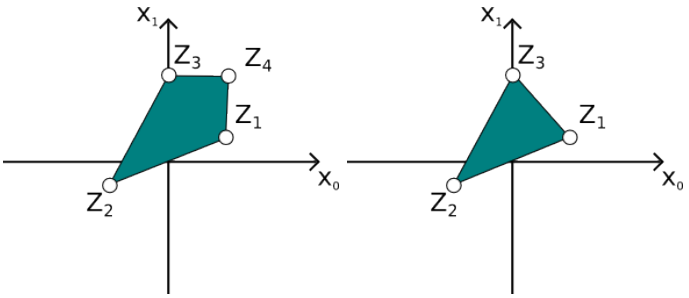
The dDMP is initialized as usual with a fixed maximum number of decision variables n. During the dynamic optimization procedure, not all of the variables are actually used in the evaluation function of the problem. The parameter $n(t)$ defines how many of the total decision variables are actually influencing the distance function at a given time t. Together with the time-dependent metric we therefore use the distance function shown in Equation 4, where only the dimensions up to $n(t)$ are actually evaluated.

For instance, we define the problem in the initialization (t=0) as a 3-dimensional problem with 3 objectives. Therefore we have $n = 3$ and $m = 3$ and the optimizer searches for a convex hull of the 3 objective-points in the 3-dimensional decision space at time $t = 0$ (i.e. $n(t = 0) = 3$). As the optimization continues, the time parameter will alter the amount of relevant variables. At time $t = 1$ we could for instance set $n(t = 1) = 2$, so that only the first two variables are taken into account when calculating the distance from a solution to each objective-point. In later Iterations, this could change again, so that $n(t)$ might be alternating between 2 and 3. It is possible to also define *which* of the variables should be the active ones (for instance through a binary vector indicating for each variable if it is active or not). For simplicity, in this work we define the number of active variables in order starting with x_0 to $x_{n(t)}$.

D. Change the Number of Objectives

Finally, we can change the number of objectives of the optimization problem. That is, we want to change for instance a 3-objective problem into a 2-objective problem by removing one of the objectives. In a DMP, this corresponds to the removal of an objective-point Z_i. As a result, the dimensionality of the objective space changes and this can have implication for the convergence and diversity behaviour of algorithms. Especially when the change goes in the other direction, and an additional objective is added, the current population of individuals might be unable to provide a good diversity in the new, larger objective space. As an example, Fig. 4 shows the change of the Pareto-optimal area (in the decision space)

of a Euclidean DMP when one objective-point, Z_4, is removed from a 4-objective problem with 2 decision variables.



(a) Four objectives (b) Three objectives

Fig. 4. Change of the Pareto-optimal solutions in the decision space with $n = 2$ variables, when a fourth objective point is added / removed.

Similar to the change of the number of variables, we implement this change of the problem in a special way in order to let algorithms treat this as a black-box problem. The maximum number of objectives is specified beforehand by the amount m given in the problem initialization. During the process, objective-points can be "removed" by simply changing their coordinates to be identical with one of the other "remaining" objective-points. In this way, the Pareto-optimal solutions set P that has to be found by the algorithm equals the one that would arise in a problem with an actual removed objective. Formally, this can be seen in Lines 2 and 3 of Equation 10.

E. Summary

Implementing all changes proposed in the last subsections, we obtain the dynamic DMP (dDMP) as follows:

$$\begin{aligned} min \quad & f(\vec{x},t) = (f_1(\vec{x},t),..,f_{m(t)}(\vec{x},t),..,f_m(\vec{x},t))^T \\ s.t. \quad & f_i(\vec{x},t) = dist(\vec{x},\vec{Z}_i(t),t) && \forall i = 1,..,m(t) \\ & f_k(\vec{x},t) = f_1(\vec{x},t) && \forall k = m(t)+1,..,m \\ & x_j \leq x_{max} && \forall j = 1,..,n \\ & x_j \geq x_{min} && \forall j = 1,..,n \end{aligned} \tag{10}$$

with

$$dist(\vec{x},\vec{Z}_i,t) := \left[\sum_{j=1}^{n(t)} |x_j - Z_{i,j}|^{p(t)}\right]^{\frac{1}{p(t)}} \tag{11}$$

and

$$Z_i(t) := \begin{pmatrix} \cos(\alpha(t)) & -\sin(\alpha(t)) \\ \sin(\alpha(t)) & \cos(\alpha(t)) \end{pmatrix} \cdot Z_i(0) + v(t) \tag{12}$$

With numbers n and m for the (maximum) numbers of variables and objectives set beforehand as in any other optimization problem, all changes that will occur during the optimization are defined by the five functions $n(t)$, $m(t)$, $\alpha(t)$, $v(t)$ and $p(t)$. In this way, we control the following properties:

- $n(t)$ is the number of active variables. It controls the dimensionality of the search space. Changing $n(t)$ changes the Pareto-optimal set of solutions as well as the optimal objective values.
- $m(t)$ is the number of active objective functions. It controls the dimensionality of the objective space. Changing $m(t)$ results in a different Pareto-optimal front as well as different Pareto-optimal solution sets.
- $\alpha(t)$ defines the rotation of the objective-points starting from its original values at $t = 0$. Rotation changes the Pareto-optimal set of solutions and the objective values (distances) of a current population to the objective points Z_i.
- $v(t)$ is the translation vector, which controls the movement of each objective point Z_i in the decision space. If all transitions are equal, i.e. $v_1(t) = v_2(t) = ... = v_m(t) = v(t)$, the Pareto-optimal set changes in term of the locations in the decision space, but the Pareto-optimal solutions still have the same objective values before and after the change.
- $p(t)$ changes the used metric. This is the most interesting change, since a change in the metric results in a change in complexity. Changing $p(t)$ can be used to make the algorithm solve easy and hard problems in an alternating fashion. For instance, from the literature we can derive that Pareto-dominance based algorithms will face difficulties with case $p = 1.0$ while the $p = 2.0$ instance is quite easy to solve [2], [17].

V. Evaluation

In this section we will briefly show the performance of some dynamic multi-objective algorithms on the proposed dynamic DMP. We will pick some of the changes introduced here and show the reaction and performance of two dynamic algorithms exemplary. Due to page limitation, we are not able to examine every possible dynamic experimentally in this paper.

For the experiments, we use two dynamic optimization algorithms. The first one is the DNSGA-II-A. It was proposed by Deb et al. [20] and is an extension of the original NSGA-II algorithm [21]. The key difference lies in the detection of changes during the optimization. In addition to the normal NSGA-II procedure, the DNSGA-II-A checks for a change in the problem by reevaluating some solutions in the current population. If their fitness values differ although the solution was not changed, the algorithm assumes a change in the optimization problem and reevaluates the whole population to obtain up-to-date objective values. In addition, a part of the population (e.g. 10%) are replaced by random new solutions, to help starting new exploration of the search space. There are two different version of the dynamic NSGA-II algorithm called DNSGA-II-A and DNSGA-II-B. The difference lies in the behaviour considering the replacement of existing solutions in case of a detected change in the problem. DNSGA-II-A replaces a part of the population with new (random) solutions, whilst DNSGA-II-B uses mutated solutions of existing problems and was described in [20] as the preferred method

for problems that are only undergoing small changes. In this work, we will solely use the DNSGA-II-A version.

The second algorithm we use is a dynamic version of the SMPSO algorithm [19]. The SMPSO is a widely-used multi-objective swarm optimizer, which was also used in previous research about the static DMP [2]. To make it applicable to dynamic environments, we equip the SMPSO with the same mechanism of change-detection and reaction as the DNSGA-II-A. This dynamic version will be called dSMPSO in this work.

We perform experiments with four different instances, using $n = 2,\ 10,\ 50,\ 100$ decision variables. For each experiment, we performed 51 independent runs with a maximum number of function evaluations of $100,000$. The necessary parameter settings are as follows. The population sizes of all algorithms are set to 100. The values for the amounts of solutions to reevaluate and to redistribute randomly are taken from [20]. That is, the share of reevaluated solutions to detect change is set to 10% of the population size and the share of randomly added solutions in case of change is set to 20%. We use Polynomial Mutation for both algorithm with a distribution index of 20.0 and a mutation probability of $1/3$. The DNSGA-II-A uses an SBX crossover with a distribution index of 20.0 and a crossover probability of 0.9. For the higher-dimensional instances ($n = 50$ and $n = 100$) the used mutation rate of $1/3$ might seem relatively high. It was set in this way to enable the algorithms to to perform a high exploration of the search space after the problem changes.

For our experiments, we chose the following 3-objective instances of the dynamic DMP. For changing the location of the objective-points, we make use of the second approach described in Subsection IV-A1 and implemented a rotation of the objective-points by defining time-dependent values for the polar-coordinates of the Z_i (See Equation 8). The used metric is $p = 2$. The following defines the time-dependent objective-points of the problem instance. For every objective-point $i \in \{1, .., m\}$:

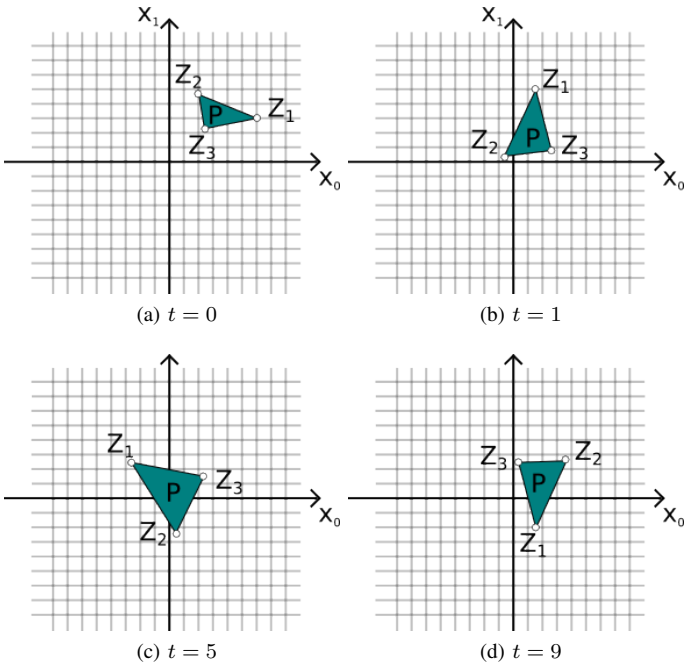
$$\begin{aligned} \alpha_{i,k}(t) &= \frac{2\pi}{3} \cdot (i-1) + \frac{\pi}{6} \cdot t \quad \forall k \in \{1, .., n-1\} \\ r_i(t) &= (5-i) - \frac{1}{6}|6-t| \\ v_{i,j} &= \frac{1}{2}|6-t| \quad \forall j \in \{1, .., n\} \end{aligned} \tag{13}$$

For the timestep we use the following formula:

$$t := \left\lfloor \frac{\#evaluations}{4000} \right\rfloor \mod 12 \tag{14}$$

Therefore, we obtain a periodic pattern with 12 different positions of the objective-points. Due to space limitations, we only show 4 out of these 12 situations exemplary in Fig. 5.

As quality indicators we use the Hypervolume rate (HVR), which is the relation between Hypervolume of the population and the Hypervolume of a sample of the true Pareto-front. The second indicator is the Spread indicator as described in [18],



(a) $t = 0$ (b) $t = 1$ (c) $t = 5$ (d) $t = 9$

Fig. 5. Four of the 12 static subproblems that form the dynamic test instance.

where a smaller Spread value indicates a better distribution of solutions.

A. Results

In this subsection we will present the results of these experiments. The focus here lies just in the understanding of the dynamic of the problem and the experiments are a showcase of how the dDMP can be used to create a dynamic instance and work with it.

TABLE I
CHANGE OF MEAN HVR BEFORE AND AFTER A CHANGE OF THE PROBLEM. VALUES IN BRACKETS ARE STANDARD ERRORS.

n	Algorithm	before Change	after Change	Δ
2	DNSGA-II-A	**0.933 (0.001)**	**0.897 (0.002)**	0.036
	dSMPSO	0.729 (0.009)	0.716 (0.009)	**0.013**
10	DNSGA-II-A	**0.806 (0.009)**	**0.377 (0.039)**	0.429
	dSMPSO	0.368 (0.024)	0.342 (0.024)	**0.026**
50	DNSGA-II-A	0.028 (0.032)	0.0 (0.007)	**0.028**
	dSMPSO	**0.121 (0.026)**	**0.084 (0.025)**	0.037
100	DNSGA-II-A	0.0 (0.003)	0.0 (0.0)	0.0
	dSMPSO	0.0 (0.028)	0.0 (0.018)	0.0

To get an impression about the adaption of the algorithm to the changes at different timesteps, we measure the HVR during the optimization each time right before and right after a change occurs. The mean HVR values before and after changes over all runs are listed in Table I. The corresponding Spread values are shown in Table II. The respective better performance before and after changes and the least loss of HVR through change are denoted in green and bold font. In addition to the values in the tables, we show in Figs. 6 to 9 the development of the HVR over time for the four different settings of n.

TABLE II
CHANGE OF MEAN SPREAD BEFORE AND AFTER A CHANGE OF THE PROBLEM. VALUES IN BRACKETS ARE STANDARD ERRORS.

n	Algorithm	before Change	after Change	Δ
2	DNSGA-II-A	0.785 (0.002)	**0.679 (0.004)**	**0.106**
	dSMPSO	**0.704 (0.011)**	0.699 (0.011)	0.005
10	DNSGA-II-A	0.735 (0.005)	**0.660 (0.014)**	**0.075**
	dSMPSO	**0.709 (0.010)**	0.708 (0.009)	0.001
50	DNSGA-II-A	**0.697 (0.009)**	**0.713 (0.010)**	**-0.016**
	dSMPSO	0.773 (0.011)	0.790 (0.010)	-0.017
100	DNSGA-II-A	**0.753 (0.011)**	**0.791 (0.014)**	-0.038
	dSMPSO	0.946 (0.012)	0.943 (0.011)	**0.003**

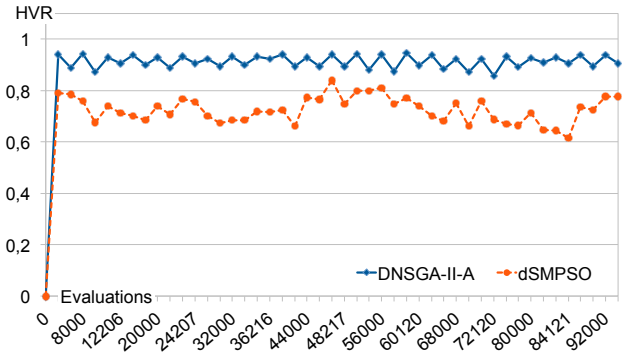


Fig. 6. Achieved HVR over time during optimization using $n = 2$ variables and the Euclidean metric ($p = 2.0$).

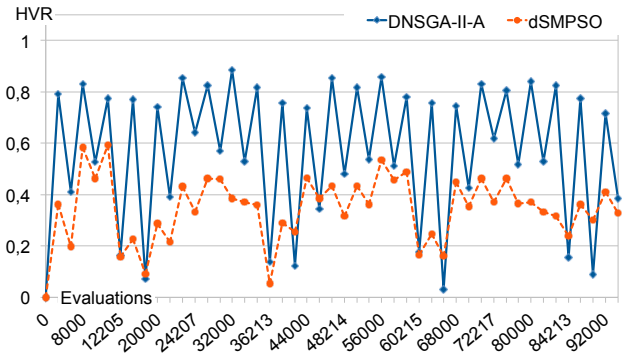


Fig. 7. Achieved HVR over time during optimization using $n = 10$ variables and the Euclidean metric ($p = 2.0$).

The figures show the data of the single run that achieved the median final HVR values.

The first interesting observation is, that throughout the optimization, the HVR values obtained before a change (see Table I) show small standard error values, especially the values of the DNSGA-II-A. This indicates that the HVR values that were achieved before a change of the problem can be achieved again until the next change happens.

It was also to be expected, that the complexity of the problem increases with higher numbers of variables. Comparing Figs. 6 and 7, we see that for $n = 2$ variables the algorithms can achieve good performance values which are only slightly disturbed by a change in the problem. On the other side, in a 10-dimensional search space (Fig. 7), we observe not only

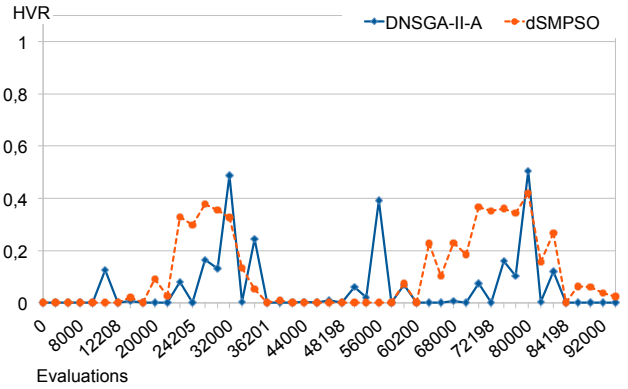


Fig. 8. Achieved HVR over time during optimization using $n = 50$ variables and the Euclidean metric ($p = 2.0$).

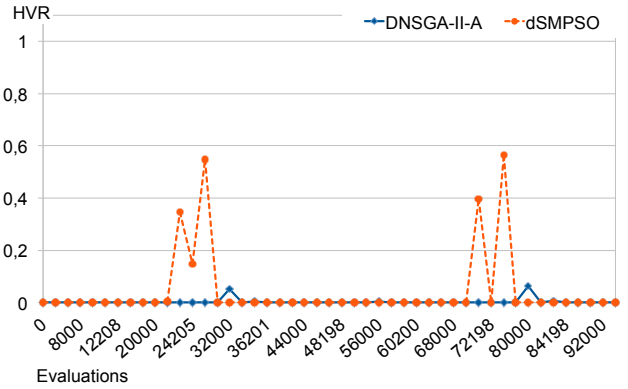


Fig. 9. Achieved HVR over time during optimization using $n = 100$ variables and the Euclidean metric ($p = 2.0$).

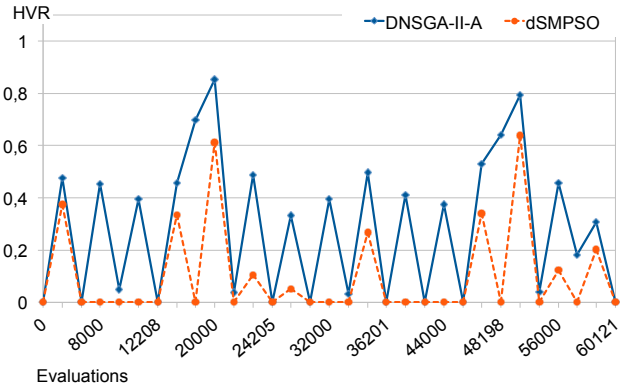


Fig. 10. Achieved HVR of the optimal solutions over time during optimization using $n = 2$ variables and the Manhattan metric ($p = 1.0$).

a lower overall performance, but more interestingly we also observe a stronger decline in performance when changes occur. To analyze this behaviour, it is easy to see that a rotation of the Pareto-optimal area in a 2-dimensional space might result in at least a part of the solutions to still be optimal (refer to the different optimal regions in Fig. 5). However, in 3-dimensional

or even higher-dimensional space, a rotation might rotate the orientation of the plane in which the optimal area lies. This can easily result in none of the previous solutions being optimal any more, thus, we see larger disturbances in the HVR.

For comparison, in Fig. 10 shows the HVR obtained over time by a similar instance of the problem using the Manhattan metric and $n = 2$ variables. We used 31 runs and slightly different initial positions and rotations to account for the special domination structure that arises in the problem. The HVR over time of the median run shown in Fig. 10 is calculated using only the Pareto-optimal solutions within the population, since the Manhattan-metric DMP is known to have a domination structure that will create many solutions close to the optimal front easily without converging further.

From the large disturbances in HVR for both algorithms in Fig. 10 as well as the overall low performance, we can see that the complexity of the DMP with $p = 1$ in the static case (refer to [2]) is also making the dynamic version a challenge. A change in the problem causes the Pareto-optimal areas to change their shape, which makes a large part of the population become non-optimal afterwards.

Due to the limitation that there exists no analytic method yet to obtain the Pareto-optimal sets for every value $1.0 \leq p \leq 2.0$, we did not test a dynamic environment where p changes during optimization yet, since the measurement of the quality indicators would be difficult. However, from the comparison of the two instances (Figs. 6 and 10) we can see that the complexity of the problem changes with the metric as claimed in Subsection IV-B. Consequently we observe in these results that also the ability of the algorithms to adapt to the changes of the problem varies.

VI. Conclusion

In this work we proposed a new benchmark function for dynamic multi-objective optimization algorithms. This dynamic Distance Minimization Problem is based on the works of the static version but makes some of its properties generic so they can be changed over time. After describing the basic principles of the problem, we described the possible dynamic changes and their implementation and effects of the Pareto-optimal areas. An advantage of using Distance Minimization Problems lies in the easy visualization of search behaviour due to geometric properties. Another interesting property compared to some other dynamic benchmarks is that the dDMP includes a parameter for changing the complexity of the problem during optimization. Our experiments show how a possible instance of the dynamic DMP can be defined and how different algorithms react to the dynamic changes.

Future work might include a method for obtaining the Pareto-optimal areas of the problem for arbitrary values of p, as well as applying this problem to other dynamic optimization methods to analyze their search behaviour.

References

[1] H. Ishibuchi, M. Yamane, N. Akedo, and Y. Nojima, "Many-objective and many-variable test problems for visual examination of multiobjective search," in *Evolutionary Computation (CEC), 2013 IEEE Congress on*, 2013, pp. 1491–1498.

[2] H. Zille and S. Mostaghim, "Properties of scalable distance minimization problems using the manhattan metric," *IEEE Congress on Evolutionary Computation (CEC)*, 2015.

[3] E. Zitzler, K. Deb, and L. Thiele, "Comparison of multiobjective evolutionary algorithms: Empirical results," *Evol. Comput.*, vol. 8, no. 2, pp. 173–195, Jun. 2000. [Online]. Available: http://dx.doi.org/10.1162/106365600568202

[4] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler, "Scalable multi-objective optimization test problems," in *Evolutionary Computation, 2002. CEC '02. Proceedings of the 2002 Congress on*, vol. 1, 2002, pp. 825–830.

[5] S. Huband, L. Barone, L. While, and P. Hingston, "A scalable multi-objective test problem toolkit," in *Evolutionary Multi-Criterion Optimization*, ser. Lecture Notes in Computer Science, C. Coello Coello, A. Hernndez Aguirre, and E. Zitzler, Eds. Springer Berlin Heidelberg, 2005, vol. 3410, pp. 280–295. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-31880-4_20

[6] M. Farina, K. Deb, and P. Amato, "Dynamic multiobjective optimization problems: test cases, approximations and applications," *IEEE Transactions on Evolutionary Computation*, 2003.

[7] J. Mehnen, G. Rudolph, and T. Wagner, "Evolutionary optimization of dynamic multiobjective functions," Universitt Dortmund, Germany, Tech. Rep., 2006.

[8] C.-K. Goh and K. C. Tan, "A competitive-cooperative coevolutionary paradigm for dynamic multiobjective optimization," *IEEE Transactions on Evolutionary Computation 13*, 2009.

[9] L. Huang, I. H. Suh, and A. Abraham, "Dynamic multi-objective optimization based on membrane computing for control of time-varying unstable plants," *Information Sciences 181*, 2011.

[10] M. Helbig and A. P. Engelbrecht, "Benchmarks for dynamic multi-objective optimisation algorithms," *ACM Computing Surveys, Vol. 46*, 2014.

[11] M. Köppen and K. Yoshida, "Many-objective particle swarm optimization by gradual leader selection," in *Adaptive and Natural Computing Algorithms*. Springer, 2007, pp. 323–331.

[12] ——, "Substitute distance assignments in nsga-ii for handling many-objective optimization problems," in *Evolutionary Multi-Criterion Optimization*. Springer, 2007, pp. 727–741.

[13] O. Schütze, A. Lara, and C. A. Coello Coello, "On the influence of the number of objectives on the hardness of a multiobjective optimization problem," *Evolutionary Computation, IEEE Transactions on*, vol. 15, no. 4, pp. 444–455, 2011.

[14] H. Ishibuchi, Y. Hitotsuyanagi, N. Tsukamoto, and Y. Nojima, "Many-objective test problems to visually examine the behavior of multiobjective evolution in a decision space," in *Parallel Problem Solving from Nature, PPSN XI*. Springer, 2010, pp. 91–100.

[15] H. Ishibuchi, N. Akedo, and Y. Nojima, "A many-objective test problem for visually examining diversity maintenance behavior in a decision space," in *Proceedings of the 13th annual conference on Genetic and evolutionary computation*. ACM, 2011, pp. 649–656.

[16] H. Masuda, Y. Nojima, and H. Ishibuchi, "Visual examination of the behavior of emo algorithms for many-objective optimization with many decision variables," in *Evolutionary Computation (CEC), 2014 IEEE Congress on*, July 2014, pp. 2633–2640.

[17] H. Zille, "Cooperative coevolution for large-scale multiobjective distance minimization problems," Master's thesis, Karlsruhe Institute of Technology (KIT), 2014.

[18] K. Deb, S. Agarwal, A. Pratap, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *Tech. Report 200001*, 2000.

[19] A. J. Nebro, J. Durillo, J. Garcia-Nieto, C. Coello Coello, F. Luna, and E. Alba, "Smpso: A new pso-based metaheuristic for multi-objective optimization," in *Computational intelligence in miulti-criteria decision-making, 2009. mcdm'09. ieee symposium on*. IEEE, 2009, pp. 66–73.

[20] K. Deb, U. B. R. N., , and S. Karthik, "Dynamic multi-objective optimization and decision-making using modified NSGA-II: A case study on hydro-thermal power scheduling," Kanpur Genetic Algorithms Laboratory (KanGAL) Indian Institute of Technology Kanpur, Tech. Rep., 2006.

[21] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *Evolutionary Computation, IEEE Transactions on*, vol. 6, no. 2, pp. 182–197, 2002.