A Scalable Many-Objective Pathfinding Benchmark Suite

Jens Weise, *Member, IEEE*, and Sanaz Mostaghim, *Senior Member, IEEE*

***Abstract*—Route planning, also known as pathfinding, is one of the key elements in logistics, mobile robotics and other applications, where engineers face many conflicting objectives. Most route planning algorithms consider only up to three objectives. In this paper, we propose a scalable many-objective benchmark problem covering most of the important features for routing applications based on real-world data. We define five objective functions representing distance, travelling time, delays caused by accidents, and two route-specific features such as curvature and elevation. We analyse several different instances for this test problem and provide their true Pareto-front to analyse the problem difficulties. Additionally, we apply four well-known evolutionary multi-objective algorithms. Since this test benchmark can be easily transferred to real-world routing problems, we construct a routing problem from OpenStreetMap data. We evaluate the three optimisation algorithms and observe that we are able to provide promising results for such a real-world application. The proposed benchmark represents a scalable many-objective route planning optimisation problem enabling researchers and engineers to evaluate their many-objective approaches.**

***Index Terms*—route finding, benchmark, many-objective optimisation, evolutionary algorithm**

I. Introduction

Optimal pathfinding is among the most challenging tasks for industrial and logistical applications [1]. Any improvement in the quality of results can have a considerable impact on many factors, such as fuel consumption and the environment. The current state-of-the-art path planning algorithms usually consider the travel time and the distance in the optimisation. However, specific applications encounter additional criteria such as the curvature of the route, the elevation (ascent), or environmental issues such as air pollution caused by fuel consumption. These criteria can profoundly influence the practicability of the solutions. For instance, for animal transportation, we need to additionally minimise the number of curves in the route (or maximise the smoothness). Reducing the length of the path can help to reduce fuel consumption while possibly increasing the travelling time. Other criteria such as the ascent of a path can be considered for heavy vehicles which can consume more fuel on such non-flat routes.

The goal of this paper is to propose a many-objective path planning problem representing five objective functions which at the same time are highly related to their real-world counterparts. This real-world problem can be considered as scalable in terms of complexity since the size of the search space can be varied. To the best of our knowledge, there is no work in the literature that considers all of these criteria simultaneously. Similar to existing navigation and route planning algorithms, we work on a graph-based approach for addressing this many-objective problem. We additionally apply the benchmark characteristics to the real-world data from OpenStreetMap in Berlin.

Our results show that this problem can be used both as a benchmark and as well as a real-world application. We additionally provide the true Pareto-front of 272 benchmark instances, their respective Pareto-sets and the code to generate specific instances of the proposed benchmark in the supplementary material[1].

The paper is structured as follows. In Section II, we provide an overview of the related works. Section III is dedicated to the many-objective pathfinding problem, the proposed encoding and the objective functions. In Section V, we provide experiments using three state-of-the-art optimisation algorithms, and in Section V-B, we transfer the benchmark and objective functions to real-world road map data. Section VI concludes the paper and gives an overview of future work.

II. Related Works

There is an extensive amount of literature in the field of route planning and pathfinding in general and especially for vehicle route planning which uses evolutionary algorithms [2], [3], [4], [5]. The most important feature concerns the solution representation, which can define the size of the search space and influence the efficiency of the algorithms.

Various representations such as graph-based [6], [7], [8], [9], [10], [11], and grid-based representations [2], [12] have been suggested for the pathfinding problem. Typically, there are two main approaches. The first is a variable-length chromosome representation which is often used in combination with the graph-based problem representation [13], [14], [15], [16]. This approach represents a solution as a list of nodes, which can be of different length when computing a path. The second approach is a fixed-length chromosome, representing the directions of travel together with a list of nodes in a graph or a list of grid cells [17], [18], [2]. Grid-based representations for pathfinding problems are shown to be very practical for evolutionary algorithms [2], [5]. Such grid representations can be refined depending on the required resolution of the problem, as a search space can be defined coarse or fine-grained. The

Manuscript received October 28, 2020; revised February 02, 2021 and April 15, 2021. This work was supported by the German Federal Ministry of Education and Research through the MOSAIK project (grant no. 01IS18070B).

J. Weise and S. Mostaghim are with the Institute for Intelligent Cooperating Systems, Otto von Guericke University Magdeburg, Germany e-mail: {jens.weise,sanaz.mostaghim}@ovgu.de.

[1]Supplementary material, including fronts, sets and code can be found here: https://ci.ovgu.de/Publications/TEVC_WM_2020-p-910.html

resolution of a problem is the granularity of its representation. Moreover, they are often used for benchmarking purposes [19], [20]. Also, they can represent the real-world problems by discretising the problem representation [21]. Grids typically consist of units with adjustable size [22]. An encoding can consist of a linked-list of units [23], the directions [2], or the coordinates of several waypoints.

It is comparatively easy to convert a grid into a graph by considering units as nodes and their contact-edges as the graph's edges. Grid-to-graph-transfer is done in several applications, e.g. the game industry when it comes to pathfinding, by superimposing a grid over an area and using graph-search algorithms [24]. The commonly used A* algorithm is an example of pathfinding on a grid which is transferred to a graph [24], [19].

In general, graph-based representations allow higher flexibility in representing real-world problems, which can be considered heterogeneous, compared to grids which are usually homogeneous. Due to this, in this paper, we present the proposed benchmark problem as a grid transferred to a graph and facilitate the methods to evaluate them on realistic graph-represented road map data.

Considering many-objective pathfinding problems, there is a limited amount of literature using evolutionary algorithms. Tozer et al. [8] provide an overview of existing approaches and use reinforcement learning to address the problem with six objective functions. Pulido et al. [9] introduce a dimensionality reduction technique in order to minimise dominance checks during the optimisation and tested their algorithm on a map from a real-world application. They extended the NAMOA* algorithm, which was first introduced by Mandow [25] and is a multi-objective extension to the well-known A* algorithm [26]. There are several existing benchmark frameworks such as [27], [28]. Additionally, there are several benchmark sets for the shortest path problem with multiple objectives, e.g. [19], [29]. The *DIMACS Implementation Challenge - Shortest Paths* [29] presents different road maps of several US-states, combining different independent data sets. There are several articles available using these graphs and a multi-objective approach, e.g. [30]. Other benchmark data sets work on grid-based approaches, e.g. [19], [8]. Usually, the data sets and graphs provided in these data sets are large, making it difficult to compute the true Pareto-front for given objectives.

In [31] three objectives are optimised by using a modified multi-objective A* algorithm, i.e. horizontal and vertical distance as well the maximal slope of a path. Machuca and Mandow use distance and time [30] as two objectives. Kanoh and Kenta additionally include a third objective called *total penalty*, in which they aggregate different negative aspects of a path on a road network [32]. In Tozer et al. [8], six objectives such as distance, signal loss to a communication station, observability of an agent, travel time, and the amount of used energy are proposed.

III. Many-Objective Pathfinding Problem

In this section, we propose a many-objective pathfinding problem which can be additionally used as a benchmark problem with a scalable size of the search space. As this benchmark aims to represent environments for pathfinding algorithms on maps, we construct the instances by defining a cartesian grid with a specific size, where the cells have the same dimensions, also known as *integer lattice*. The variable properties of the benchmark influence the properties of each cell in the lattice.

A. Benchmark problem construction

The multi-objective pathfinding problem can be defined as a network-flow problem [33], [9]. The goal is to find a set of optimal paths $P^* = \{p_1, \cdots, p_L\}$ in a graph $G(V, E)$ from a starting node $n_S \in V$ to a pre-defined end node $n_{End} \in V$, i.e., $p_i = (n_S, \cdots, n_{End})$. Before constructing the problem-related graph, we model a grid which is used as a map for the pathfinding problem. We assume to have a two-dimensional search space defined by a given size (i.e. size of the map) denoted by the range $[1, x_{max}]$ in x-direction and $[1, y_{max}]$ in y-direction, $x, y \in \mathbb{N}$. This search space is divided into grid cells which define the resolution of the path planning and therefore, the size of the search space. Eventually, the grid has x_{max} number of grid cells in x-direction, and y-direction accordingly. We define different types of grid cells denoted by the position (x, y) which impose constraints on the velocity of movements indicated by $v_{max}(x, y)$ representing different road types as well as obstacles (denoted by $g_{LA}(x, y)$ and $g_{CH}(x, y)$ in supplementary materials) where a movement cannot occur. The cells with a velocity of zero define infeasible areas which can add additional non-linear constraints.

We furthermore define an elevation function $h(x, y)$ with a variable number of hills which can be defined by either using a peak-function or a combination of hill functions. This elevation function can be mapped to the cells in the map. Two more features concern the neighbourhood and backtracking. The neighbourhood restricts the movement to the possible neighbour cells to which an agent can move. We use the 2^k-neighbourhood similar to [34]. In this case, $k = 2$ means that it is possible to go to one of the four neighbours, known as the *von Neumann neighbourhood*, located in the cardinal directions, where $k = 3$ defines eight possible neighbours, taking the diagonal cells into account, known as the *Moore neighbourhood*. The backtracking property of the benchmark defines if an agent can go backwards or only forward. For instance, if backtracking is allowed and the goal is to go from the north-west corner of the grid to the southern-east one, the agent can go in any direction specified by the 2^k-neighbourhood from any cell on a certain path. If backtracking is not allowed, the agent can only move in the directions of east, south and south-east (if $k = 3$). An 8-neighbourhood with enabled backtracking is also known as *king-moves*, derived from chess. A summary of the above adjustable features are shown in Table I.

In the following, we propose a graph-based representation of the benchmark grid. Therefore, we describe all objectives for the evaluation of a *solution* represented as a path N of variable length k consisting of a list of adjacent nodes in a graph $G = (V, E)$: $N = (n_i, n_{i+1}, \cdots, n_k) = p_i$, where $n_i =$

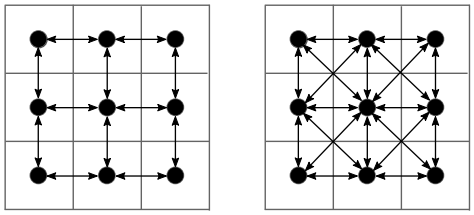


Fig. 1. Superimposed graphs on grids for K2 (left) and K3 (right) instances

TABLE I
ADJUSTABLE PROPERTIES OF THE PROPOSED BENCHMARK

Property	Values
Size	$\{x, y\}_{max} \in \mathbb{N}$, $1 < \{x, y\}_{max}$
Movement per cell	v_{max}
Expected delay	$delay(n_i, n_{i+1})$
Elevation Function	$nh \in \{1, 2, 3, M\}$
Neighbourhood	2^k, $k \in \{2, 3\}$
Backtracking	$\{True, False\}$

n_S, and $n_k = n_{End}$. However, for the evaluation on the grid (as described above), the nodes n_i can be replaced by their respective coordinates (x_i, y_i).

To transfer the grid to its corresponding graph, each cell c_i of the grid with its respective coordinates (x_i, y_i) is considered as a node in the graph. In our implementation, we assign properties to the graph's elements in the form of key-value pairs, utilising the property graph data model [35]. Therefore, we can assign the coordinates as a property to each node, making it possible to evaluate the objectives. Additionally, the various cell types, velocity constraints, characteristics about obstacles, and elevation values are assigned to the properties of the node. Depending on the 2^k-neighbourhood and backtracking property, the corresponding nodes are connected to their respective neighbours using edges. The resulting graph is also known as *lattice graph*. Figure 1 shows an example of the transfer from a grid to a graph.

B. Objective functions

In this section, we will define five objective functions, by which a solution path N is evaluated.

Objective 1: Euclidean length. The Euclidean length represents the distance between the start n_S and the end n_{End} of a path. It is calculated by the sum of the Euclidean distances $d(n_{i-1}, n_i)$ between the neighbouring vertex pairs n_{i-1} and n_i in a solution path N as follows:

$$f_1(N) = \sum_{i=1}^{K-1} d(n_i, n_{i+1}) \tag{1}$$

We consider that $i = 1$ corresponds to the starting point n_S and the last node of a path n_K maps to the endpoint denoted n_{End}. Figure 2 illustrates an example. In real-world applications, this objective can be additionally used to estimate fuel consumption.

Objective 2: Expected delays. The second objective is meant to measure the expected delay in a given path. In

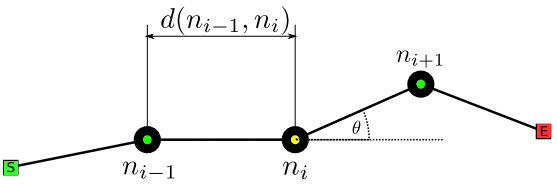


Fig. 2. Objectives (1) and (5) on an example path modelled by a graph

real-world applications, delays are caused by accidents or traffic. Therefore, a delay is a likelihood of having an accident or encountering other blockages on each node of the path. However, in our proposed approached we do not take draws from the probability distribution, making it deterministic. The expected delay per path segment between the nodes n_i and n_{i+1} is defined by the differences between the corresponding velocity values of the two adjacent nodes. Our proposed second objective f_2 calculates the sum of $delay$ for all the edges on a given path N:

$$f_2(N) = \sum_{i=1}^{K-1} delay(n_i, n_{i+1}) \tag{2}$$

Objective 3: Elevation. The aggregated ascent of a solution path is represented by the third objective. Our proposed benchmark contains various possibilities for defining the elevation function $h(n_i)$ which is defined on a node n_i. The ascent is calculated between two nodes in the graph $e(n_i, n_{i+1})$. and the third objective $f_3(N)$ is the sum of the elevations between all the nodes in a path N:

$$\begin{aligned} f_3(N) &= \sum_{i=1}^{K-1} e(n_i, n_{i+1}) \\ e(m, n) &= \begin{cases} h(n) - h(m), & \text{if } h(n) > h(m) \\ 0, & \text{otherwise} \end{cases} \end{aligned} \tag{3}$$

This objective can represent the amount of fuel consumption in a real-world application.

Objective 4: Traveling time. The fourth objective represents the traveling time. For this purpose, we utilise the average velocity of two subsequent nodes defined by $\frac{v_{max}(n_i)+v_{max}(n_{i+1})}{2}$ for each node n_i and use the length of the path utilised in Objective 1:

$$f_4(N) = \sum_{i=1}^{K-1} \frac{2d(n_i, n_{i+1})}{v_{max}(n) + v_{max}(n_{i+1})} \tag{4}$$

Objective 5: Smoothness. The smoothness, or curvature, of a path is modelled in the fifth objective. We measure smoothness by calculating the angle between three nodes on a path, as shown in Figure 2. The angle θ is obtained by extending the line between two nodes and measuring the angle to the third node. Similar to [36], [14], we invert $a \cdot b = \|a\|\|b\|cos(\theta)$:

$$f_5(N) = \sum_{i=2}^{K-1} arccos\left(\frac{\overrightarrow{n_i n_{i-1}} \cdot \overrightarrow{n_{i+1} n_i}}{|\overrightarrow{n_i n_{i-1}}| \cdot |\overrightarrow{n_{i+1} n_i}|}\right) \tag{5}$$

Since we intend to minimise the objective values, the smaller smoothness value represents a more straight path.

IV. Benchmark Test Suite

In the following, we propose various examples for a test suite by selecting specific features for the defined variables of the benchmark. We set $n_S = (1, 1)$ as the start and $n_{End} = (x_{max}, y_{max})$ as the end nodes. In this way, a path starts in the north-western and ends in the south-eastern corners. We set 4 various kinds of cells with velocity values v_{max} of 130, 100, 50 and 0. As for obstacle cells with $v_{max} = 0$, we propose two different forms: 1) the chequerboard pattern is designed to simulate block-like environments, and 2) the lake obstacle denotes a larges region which is not passable (Figure 3). For the chequerboard obstacles, we define every second cell to be an obstacle in both x and y directions. The lake obstacle is defined as a circle on the grid. The circle radius is defined by a fraction of the x-size of the grid. We represent the checkerboard and lake obstacles as a variant of the square wave function and circle function, respectively (Equations are provided in the supplementary materials). In the tested instances, the lake obstacles are defined by a radius of $x_{max}/4$. Figures 3b to 3c show the two obstacle types on an example instance of the benchmark problem. Figure 3a shows an example instance of size 20.

The corresponding equations are provided in the supplementary materials.

As for the elevation, we take four hill functions in the domain $[-3, 3]$ which will be scaled when applied to the grid with cell coordinates (x, y) represented by the node n in the path segment. To determine the corresponding height value $h(x, y)$, the two cell coordinates have to be scaled to the interval $[-3, 3]$, hence $[\{1, 1\}, \{x_{max}+1, y_{max}+1\}] \rightarrow [-3, 3]$ and $(x, y) \rightarrow (x_s, y_s), \{x_s, y_s \in \mathbb{R} | -3 \leq x_s, y_s \leq 3\}$. In the equation, we refer to (x_s, y_s) to represent the scaled coordinates.:

$$\begin{aligned} h_m(x_s, y_s) =& 3(1 - x_s)^2 e^{-x_s^2 - (y_s+1)^2} - 10e^{-x_s^2 - y_s^2} \\ & (-x_s^3 + x_s/5 - y_s^5) - 1/3e^{-(x_s+1)^2 - y_s^2} \\ h_1(x_s, y_s) =& 5e^{-(x_s+1.5)^2 - (y_s+1.5)^2} \\ h_2(x_s, y_s) =& 5e^{-(x_s-1.5)^2 - (y_s-1.5)^2} \\ h_3(x_s, y_s) =& 5e^{-(x_s-1.5)^2 - (y_s+1.5)^2} \end{aligned} \tag{6}$$

We choose these functions to represent different height settings on the grid. Equation h_m also known as *peaks*-function has various hills and valleys. Since this function is defined in the interval of $[-3, 3]$, we define the other three functions in the same interval. Each of the other three equations represents a hill on the landscape. In the supplementary material in Section A-C, these functions' linearly combinations are depicted. By combining them, various elevation characteristics of the problem instances can be achieved. Eventually, an instance can have h_m *or* a linear combination of the others as its elevation function. Therefore, we define h as:

$$h(x, y) = \begin{cases} \sum_{i=1}^{nh} h_i, & \text{if } nh \in \{2, 3\} \\ h_3, & \text{if } nh = 1 \\ h_m, & \text{if } nh = M \end{cases} \tag{7}$$

For the third objective, we aggregate positive slopes, as we want to focus on flat routes. Taking also negative elevations into account can result in a path containing a hill with a steep gradient which cannot be beneficial for a bulky transportation. The fourth objective, expected delay, is defined by v_{max} of two subsequent cells (refer to supplementary materials Equations (10) and (11).)

All these variations of the properties are used in the name of a benchmark instance. The name starts with *ASLETISMAC* for the five objectives to be minimised (Ascent, Length, Time, Smoothness and Accidents (expected delay)), then the obstacle type, followed by the size in X and Y directions, then the elevation function is represented (PM stands for the peaks-function h_m and the combination is set to Pnh), followed by the 2^k-neighbourhood and the backtracking property (B followed by T for True or F for False). For example, *ASLETISMAC_CH_X10_Y10_P1_K2_BF* defines an instance with the chequerboard obstacles, sized $10x10$, $nh = 1$ as the elevation function (one hill), four possible neighbours (K2), but no backtracking (BF). For the values of delays (caused by accidents) in the second objective, we refer to real-world statistical data (see Equation (2))[2]. We adopt the likelihood of encountering an accident from real-world data, depending on the v_{max} of a certain cell. For instance, it is much more likely to have an accident when driving on streets located in a city, i.e. with a lower v_{max}. Therefore, we assume a smaller likelihood of encountering an accident with higher velocities. We also assume a large likelihood when the type of street changes, e.g. going over an access road or an exit road.

A. Obtaining the true Pareto-Front

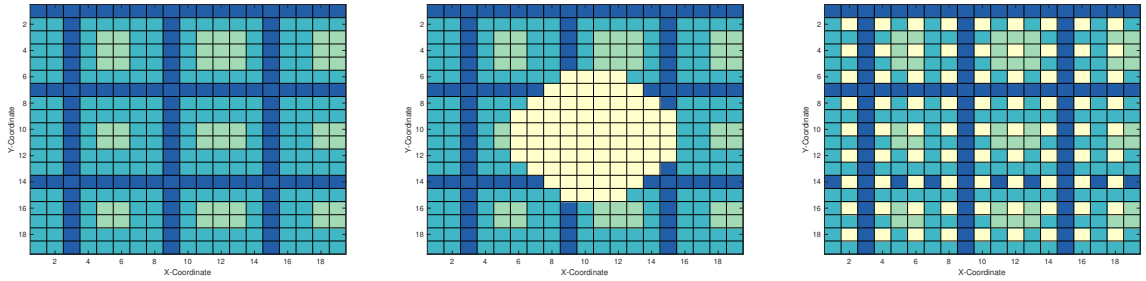
We have performed an exhaustive search on 272 benchmark instances with different obstacle types, sizes, elevation-functions, neighbourhood metrics. In order to obtain the fronts, we performed a depth-first search (DFS) from the cell at the northern-west corner to the south-east corner cell. The larger the instances are, the longer the DFS takes to complete. The most complex in terms of the number of possible paths, which we evaluated, is the instance *ASLETISMAC_NO_X14_Y14_PX_K3_BF*, that has a size of 14x14, 4-neighbourhood and no backtracking. For this instance, there are $1,409,933,619$ possible paths.

The number of possible paths are represented by specific integer sequences (visible at oeis.org) and is shown in Table III in supplementary materials.

B. Benchmark Characteristics

The proposed benchmark has several specific characteristics. Regarding the decision space, we can define a fixed length or a variable length encoding of solutions. Fixed encodings are suggested especially for the *K2,BF* instances, as the allowed paths have the same length $f_1(N)$, i.e. $f_1(N) = ((x_{max} - 1) + (y_{max} - 1))$. Using a variable-length approach can represent the problem as a combinatorial one. For this

[2]https://www.destatis.de/EN/Themes/Society-Environment/Traffic-Accidents/_node.html



(a) Three different types of cells (b) *lake* obstacle (c) *checkboard*-obstacles

Fig. 3. Examples of grid cell properties (dark to light colours represent high to low speed values)

purpose, we can use graph, real-value or integer-value representations. In this case, the true Pareto-fronts of the test instances are disconnected and degenerate due to the discreet search space. Also, the fronts are irregular, and the different objectives have different scales. An interesting characteristic of this benchmark regards the fact that similar paths on the grid are not necessarily close in the objective space, implying that paths which are different in the majority of their nodes can lead to similar objective values. In the *K2,BF* instances, the challenges for algorithms depend on the chosen representation to find a feasible path, as the ratio of infeasible to feasible solutions is relatively high. In the supplementary material, Figures 10 to 14 show two examples of obtained true Pareto-sets and fronts, as well as algorithmic results.

V. Experiments

In the experiments, we aim to investigate the degree of the difficulty of the proposed benchmark problem. We apply four different state-of-the-art evolutionary algorithms to several instances to evaluate the complexity of the benchmark. Furthermore, we present a custom mutation operator, which can operate on a variable-length chromosome consisting of a list of nodes.

In our proposed benchmark, we consider a solution to be a sequence of nodes $N = (n_1, \cdots, n_k)$ with a variable-length k. We take this representation for the encoding in evolutionary algorithms. The variable-length chromosome poses difficulties for the algorithms but can be very efficient when using realistic data since intersections and endpoints are not homogeneously distributed, and paths usually have different lengths. This representation has been used by [37], [38], [39] and studied by [40].

We use a one- or two-point cross-over for this encoding as follows. If two selected solutions have intersection points except for the start and end nodes, these points can be used as possible cut-off points. If there are fewer than two intersections, we use a one-point cross-over. Additionally, we define the so-called *perimeter mutation operator*. From a given path which is to be mutated, we take two arbitrary points within a maximum network distance $d_{max} = \frac{|N|}{2}$ and compute their middle point. Then we search for a random point within a maximum distance of r_{max}, using an R-Tree index, which was generated upfront [41]. We perform a random-search (local search) from the first and second points to it. Depending on the benchmark instance, we either consider all neighbouring nodes within the radius in positive cardinal and diagonal directions (instances of type *K3,BF*) or a subset of them: nodes in positive cardinal directions for *K2,BF*.

In the experiments, we use the NSGA-II [42], NSGA-III [43] and DIR-enhanced NSGA-II (d-NSGA-II) [44] algorithms. The d-NSGA-II uses a diversity indicator based on reference vectors [44], making it suitable for many-objective optimisation problems. Additionally, we used an indicator-based algorithm, i.e. the $\mathrm{I}_{SDE}+$ algorithm [45]. For all four algorithms, we set the population size to 212 as in the original NSGA-III study. We set the probabilities for cross-over and mutation to 0.8 and 0.2, the number of divisions for NSGA-III to $p = 6$, maximum number of generations to 500, and all for 31 runs for statistical analysis. The task of the pathfinding algorithm is to find a path from the north-west corner to the south-east corner.

To compare the algorithms, we calculated the IGD^+ indicator [46], [47]. The results are compared and tested for statistical significance using the non-paremetric Kruskal-Wallis test and Bonferroni correction for multiple independent samples, as suggested by Knowles et al. [48]. The null-hypothesis states that that the distributions of the four samples have equal medians. Statistical significance of the differences between the performance is assumed for a p-value smaller than 0.01.

A. Results

In the first part of our analysis, we count the number of successful runs in which the algorithms could obtain the entire Pareto-front. A front is found if the IGD^+ is 0 in all 31 runs on the algorithms. Given 272 valid instances, NSGA-II, NSGA-III, d-NSGA-II and $\mathrm{I}_{SDE}+$ were not able to find the complete true Pareto-fronts for 234, 233, 240 and 221 instances. This indicates the difficulty of the benchmark for specific instances. For 15 instances, the algorithms did not find a result. This outcome occurred mostly on the small *X3_Y3* instances. The reason for this is the customised operators which can fail on relatively short paths. It can occur that the mutation operator will not find a suitable node in the given radius. In the future, we want to make the operators more robust to any length of the paths. After running the experiments, we obtained complete results for 257 instances. By complete we mean that we

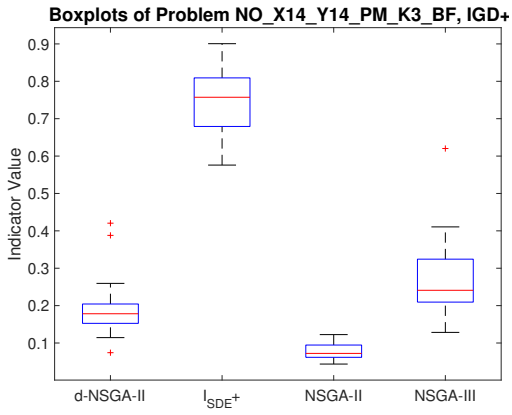


Fig. 4. Obtained IGD$^+$ Values for the instance ASLETISMAC_NO_X14_Y14_PM_K3_BF

TABLE II
WINS, LOSSES AND TIES OF EACH ALGORITHM WITH STATISTICAL SIGNIFICANCE AT $p < 0.01$, BONFERRONI CORRECTION APPLIED, IGD$^+$ INDICATOR.

	d-NSGA-II	I$_{SDE}$+	NSGA-II	NSGA-III
Wins	3	9	164	7
Losses	174	196	31	191
Ties	80	52	62	67

obtained the results from all 31 runs for an instance for all four algorithms. Figure 4 shows the obtained IGD$^+$ values for the instance ASLETISMAC_NO_X14_Y14_PM_K3_BF for which none of the algorithms found the whole Pareto-front, indicating the complexity of the problem. We observe that NSGA-II obtains the best result, even if NSGA-II is not usually thought to be the best option for many-objective problems. In Table II, the wins, losses and ties are shown for each of the used algorithms.

Overall, NSGA-II performed the best in the IGD$^+$ indicator on the majority of instances (statistically significant difference for $p < 0.01$, see Figure 9), which can be due to the crowding distance estimation to maintain diversity which is beneficial to irregular Pareto-fronts [44]. However, the results of the I$_{SDE}$+ algorithms indicate, that more algorithms of this class have to be tested on the benchmarks, as it has the highest number of completely solved instances. Five other instances, their Pareto-Set and the result set of the four algorithms are presented in the supplementary material in Figures 10 to 14. The I$_{SDE}$+ algorithm shows the most diverse results in the decision space. When analysing the algorithms' progress, we often saw in the *K2,BF* instances that some algorithms converged to paths going only down and then right. Therefore, we conclude that it can be challenging for algorithms to explore these instances' search space, as they can fall into local optima. The proposed benchmark suite generates instances, in which closeness of paths does not reflect closeness in objective space. In conclusion, size, neighbourhood, and backtracking increase the search-space size, and conversely, by changing the two latter to a value which decreases the search-space, they also increase the ratio of infeasible to feasible solutions. The convergence to local optima can be observed in Figure 14 in the supplementary material. From a visual perspective it seems that the I$_{SDE}$+ algorithm is less prone to these challenges.

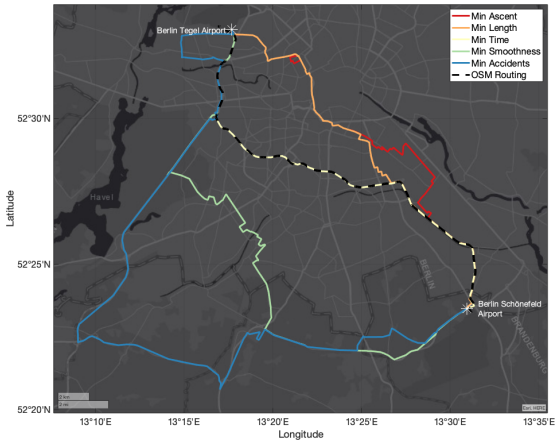


Fig. 5. Map of Berlin showing the best path in terms of each objective. Min Ascent, Min Length, Min Time, Min Smoothness, Min expected delay, the dashed black line represents the path from the original OpenStreetMap Routing Service

B. Real-World Data

In the following, we aim to transfer the problem from the proposed benchmark to a real-world application. We use the data on the map of Berlin and compute a set of paths between the two airports *Berlin-Tegel* and *Berlin-Schönefeld*. For this purpose, we use OpenStreetMap data which is imported and converted to an undirected graph via the *osmnx* library [49]. We simplify the network by removing nodes which do not represent an intersection. The resulting graph has 63731 vertices and 84912 edges. For merged edges, we took the maximum values of the merged partners and aggregated the distances. Due to this, our computed path is an approximation but can be used to analyse the algorithm's performance on real-world data. Figure 5 shows the layout of the map and depicts the start and endpoint.

Since this is a real-world problem, we do not know the true Pareto-front. To approximate the performance of the algorithms, we combined all results from all four algorithms and all 31 runs and calculated the non-dominated solution set. We obtained 1422 non-dominated solutions. Figure 5 shows a subset of the obtained non-dominated solutions and the path obtained from the OpenStreetMap routing service. For clarity, we do not illustrate the whole set but only five non-dominated paths from one airport to the other, representing the best solution per objective. It is visible that the paths have differences. Furthermore, the paths with the fewest number of accidents are mostly going over highways, indicating that the algorithms could explore the search space. Interestingly, our obtained path with the least time is the same as that obtained from the OSM routing service. All depicted paths could be recommended to a hypothetical driver, representing different possible requirements. The blue route is most likely the most reliable one since it contains the least expected delay, albeit being comparably long. Also, the red route is suitable for vehicles with less power. With the obtained reference from

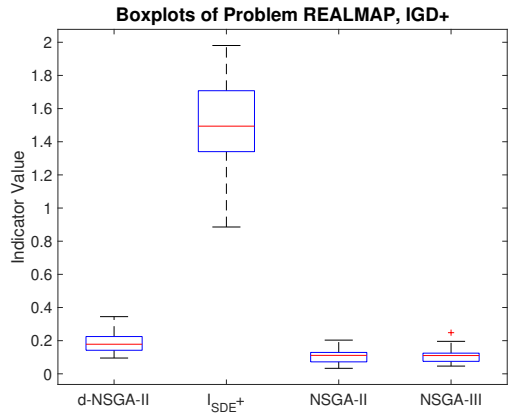


Fig. 6. Obtained IGD$^+$ values on Real-world problem

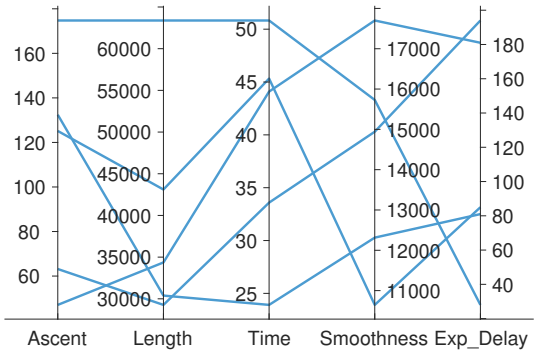


Fig. 7. Parallel coordinates plot of the best paths for each objective

all runs, we were able to calculate the IGD$^+$ indicator for the three algorithms. Figure 6 shows the respective values of the obtained results and Figure 7 shows the parallel coordinates plot of the best solutions per objective. In this experiment, NSGA-III obtained the best median; however, it significantly outperforms d-NSGA-II and I$_{SDE}+$ only.

The experiments show that while the NSGA-II performed the best on the majority of the benchmark instances, the NSGA-III is at least equally good on the real-world example. The artificial instances distinguish from the real-world example as they are ordered as a grid, while real-world data is usually more heterogeneous. An algorithm's performance can depend on the underlying structure. The artificial instances reflect the properties of real-world street networks to a certain extent while being scalable and variable.

VI. Conclusion

In this paper, we present a scalable many-objective pathfinding benchmark problem representing a real-world related navigation problem on actual map data. The benchmark is scalable and can be used to analyse many-objective optimisation techniques for path and route planning and navigation. Different obstacle types, as well as elevation functions, neighbourhoods and backtracking properties, can be adjusted according to the required complexity. We proposed five objective functions for the benchmark related to real-world goals when planning a route. Furthermore, we obtained the true Pareto-fronts for several benchmark instances which we also provide in the supplementary material. Additionally, we used three existing evolutionary algorithms to minimise five objectives and compared the results with the obtained true Pareto-fronts of several benchmark instances. Moreover, we transferred the benchmark's characteristics to real-world data by adding further information to an obtained OpenStreetMap data graph. We also applied the algorithms with the same parameters and could obtain promising results.

In the future, other real-world road characteristics will be included in the benchmark suite, such as bridges, one-way or two-way streets or different distributions of certain road types, reflecting the hierarchical properties of real-world examples. In addition, we will work on advanced algorithms and operators. Besides, we aim to analyse more extensive and complex instances of the problem, specifically the instances with enabled backtracking. As this will increase the search space by several magnitudes, we also intend to investigate more sophisticated methods to obtain the true Pareto-fronts of more complex instances.

References

[1] P. Toth and D. Vigo, *Vehicle routing: problems, methods, and applications*. SIAM, 2014.

[2] F. Ahmed and K. Deb, "Multi-objective optimal path planning using elitist non-dominated sorting genetic algorithms," *Soft Computing*, vol. 17, no. 7, pp. 1283–1299, 7 2013.

[3] O. Castillo, L. Trujillo, and P. Melin, "Multiple objective genetic algorithms for path-planning optimization in autonomous mobile robots," *Soft Computing*, vol. 11, no. 3, pp. 269–279, 2 2007.

[4] M. Alajlan, A. Koubaa, I. Chaari, H. Bennaceur, and A. Ammar, "Global path planning for mobile robots in large-scale grid environments using genetic algorithms," in *2013 International Conference on Individual and Collective Behaviors in Robotics (ICBR)*, no. 1. IEEE, 2013, pp. 1–8.

[5] F. Ahmed and K. Deb, "Multi-objective path planning using spline representation," *2011 IEEE International Conference on Robotics and Biomimetics, ROBIO 2011*, no. 2011010, pp. 1047–1052, 2011.

[6] L. Changan, Y. Xiaohu, L. Chunyang, and L. I. Guodong, "Dynamic Path Planning for Mobile Robot Based on Improved Genetic Algorithm," *Chinese Journal of Electronics*, vol. 19, no. 2, 2010.

[7] Z. Qiongbing and D. Lixin, "A new crossover mechanism for genetic algorithms with variable-length chromosomes for path optimization problems," *Expert Systems with Applications*, vol. 60, pp. 183–189, 10 2016.

[8] B. Tozer, T. Mazzuchi, and S. Sarkani, "Many-objective stochastic path finding using reinforcement learning," *Expert Systems with Applications*, vol. 72, pp. 371–382, 4 2017.

[9] F.-J. J. Pulido, L. Mandow, and J.-L. L. Pérez-De-La-Cruz, "Dimensionality reduction in multiobjective shortest path search," *Computers & Operations Research*, vol. 64, pp. 60–70, 7 2015.

[10] M. Rajabi-Bahaabadi, A. Shariat-Mohaymany, M. Babaei, and C. W. Ahn, "Multi-objective path finding in stochastic time-dependent road networks using non-dominated sorting genetic algorithm," *Expert Systems with Applications*, vol. 42, no. 12, pp. 5056–5064, 2015.

[11] J. Weise, S. Benkhardt, and S. Mostaghim, "A Survey on Graph-based Systems in Manufacturing Processes," in *2018 IEEE Symp. Series on Computational Intelligence*, pp. 112–119.

[12] K. Yakovlev, E. Baskin, and I. Hramoin, "Grid-Based Angle-Constrained Path Planning," in *Lecture Notes in Computer Science*, 2015, vol. 9324, pp. 208–221.

[13] A. Elshamli, H. A. Abdullah, and S. Areibi, "Genetic algorithm for dynamic path planning," in *Canadian Conference on Electrical and Computer Engineering 2004*, vol. 2. IEEE, 2004, pp. 677–680.

[14] J. Hu, Q. Zhu, H. Jun, and Z. Qingbao, "Multi-objective Mobile Robot Path Planning Based on Improved Genetic Algorithm," in *2010 International Conference on Intelligent Computation Technology and Automation*, vol. 2. IEEE, 2010, pp. 752–756.

[15] S. Mittal, K. Deb, Shashi Mittal, and Kalyanmoy Deb, "Three-dimensional offline path planning for UAVs using multiobjective evolutionary algorithms," in *2007 IEEE Congress on Evolutionary Computation*, pp. 3195–3202.

[16] M. Davoodi, F. Panahi, A. Mohades, and S. N. Hashemi, "Multi-objective path planning in discrete space," *Applied Soft Computing*, vol. 13, no. 1, pp. 709–720, 1 2013.

[17] E. Besada-Portas, L. de la Torre, A. Moreno, and J. L. Risco-Martín, "On the performance comparison of multi-objective evolutionary UAV path planners," *Information Sciences*, vol. 238, pp. 111–125, 7 2013.

[18] H. Qu, K. Xing, and T. Alexander, "An improved genetic algorithm with co-evolutionary strategy for global path planning of multiple mobile robots," *Neurocomputing*, vol. 120, pp. 509–517, 11 2013.

[19] N. R. Sturtevant, "Benchmarks for Grid-Based Pathfinding," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 4, no. 2, pp. 144–148, 2012.

[20] S. Koceski, S. Panov, N. Koceska, P. B. Zobel, and F. Durante, "A Novel Quad Harmony Search Algorithm for Grid-Based Path Finding," *International Journal of Advanced Robotic Systems*, vol. 11, no. 9, p. 144, 9 2014.

[21] B. Anguelov, "Video Game Pathfinding and Improvements to Discrete Search on Grid-based Maps," Ph.D. dissertation, 2011.

[22] A. R, "Path Finding Solutions For Grid Based Graph," *Advanced Computing: An International Journal*, vol. 4, no. 2, pp. 51–60, 3 2013.

[23] J. Xiao and Z. Michalewicz, "An Evolutionary Computation Approach to Robot Planning and Navigation," *Soft Computing in Mechatronics*, vol. 32, pp. 117–141, 1999.

[24] P. Yap, "Grid-Based Path-Finding," in *Lecture Notes in Computer Science*, 2002, vol. 2338, pp. 44–55.

[25] L. Mandow and J. L. P. De La Cruz, "Multiobjective A * search with consistent heuristics," *Journal of the ACM*, vol. 57, no. 5, pp. 1–25, 6 2010.

[26] P. E. Hart, N. J. Nilsson, and B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[27] J. E. Fieldsend, T. Chugh, R. Allmendinger, and K. Miettinen, "A feature rich distance-based many-objective visualisable test problem generator," in *Proc. of the Genetic and Evolutionary Computation Conference*. New York, USA: ACM, 7 2019, pp. 541–549.

[28] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler, "Scalable Test Problems for Evolutionary Multiobjective Optimization," in *Evolutionary Multiobjective Optimization*. Springer-Verlag, pp. 105–145.

[29] C. Demetrescu, A. Goldberg, and D. Johnson, "The shortest path problem : ninth DIMACS implementation challenge," 2009.

[30] E. Machuca and L. Mandow, "Multiobjective heuristic search in road maps," *Expert Systems with Applications*, vol. 39, no. 7, pp. 6435–6445, jun 2012.

[31] M. S. Rahaman, Y. Mei, M. Hamilton, and F. D. Salim, "CAPRA: A contour-based accessible path routing algorithm," *Information Sciences*, vol. 385-386, pp. 157–173, apr 2017.

[32] H. Kanoh and K. Hara, "Hybrid genetic algorithm for dynamic multi-objective route planning with predicted traffic in a real-world road network," in *Proc. of the 10th annual conference on Genetic and evolutionary computation*. New York, USA: ACM Press, 2008, p. 657.

[33] A. Raith and M. Ehrgott, "A comparison of solution strategies for biobjective shortest path problems," *Computers & Operations Research*, vol. 36, no. 4, pp. 1299–1331, 4 2009.

[34] R. Stern, N. R. Sturtevant, A. Felner, S. Koenig, H. Ma, T. T. Walker, J. Li, D. Atzmon, L. Cohen, T. K. S. Kumar, E. Boyarski, R. Bartak, R. Bart, R. Bartak, T. K. Satish Kumar, E. Boyarski, and R. Barták, "Multi-Agent Pathfinding: Definitions, Variants, and Benchmarks," no. SoCS, pp. 151–158, 6 2019.

[35] M. A. Rodriguez and P. Neubauer, "Constructions from dots and lines," *Bulletin of the American Society for Information Science and Technology*, vol. 36, no. 6, pp. 35–41, aug 2010.

[36] B. K. Oleiwi, H. Roth, and B. I. Kazem, "Modified Genetic Algorithm based on A* Algorithm of Multi Objective Optimization for Path Planning," *Journal of Automation and Control Engineering*, vol. 2, no. 4, pp. 357–362, 2014.

[37] C. Lamini, S. Benhlima, and A. Elbekri, "Genetic algorithm based approach for autonomous mobile robot path planning," *Procedia Computer Science*, vol. 127, pp. 180–189, 2018.

[38] A. Tuncer and M. Yildirim, "Dynamic path planning of mobile robots with improved genetic algorithm," *Computers and Electrical Engineering*, vol. 38, no. 6, pp. 1564–1572, 11 2012.

[39] Q. Li, W. Zhang, Y. Yin, Z. Wang, G. Liu, and Z. Wang Guangjun Liu, "An Improved Genetic Algorithm of Optimum Path Planning for Mobile Robots," in *6th International Conference on Intelligent Systems Design and Applications*, vol. 2. IEEE, 2006, pp. 637–642.

[40] L. Beke, M. Weiszer, and J. Chen, "A Comparison of Genetic Representations for Multi-objective Shortest Path Problems on Multigraphs." Springer International Publishing, 2020, vol. 8, no. 2010, pp. 35–50.

[41] A. Guttman, "R-trees: a dynamic index structure for spatial searching," *ACM SIGMOD Record*, vol. 14, no. 2, pp. 47–57, jun 1984.

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

[43] K. Deb and H. Jain, "An Evolutionary Many-Objective Optimization Algorithm Using Reference-Point-Based Nondominated Sorting Approach, Part I: Solving Problems With Box Constraints," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 4, pp. 577–601, 2014.

[44] X. Cai, H. Sun, and Z. Fan, "A diversity indicator based on reference vectors for many-objective optimization," *Information Sciences*, vol. 430-431, pp. 467–486, 3 2018.

[45] T. Pamulapati, R. Mallipeddi, and P. N. Suganthan, "ISDE+ - An Indicator for Multi and Many-Objective Optimization," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 2, pp. 346–352, apr 2019.

[46] H. Ishibuchi, H. Masuda, and Y. Nojima, "A Study on Performance Evaluation Ability of a Modified Inverted Generational Distance Indicator," in *Proc. of the 2015 on Genetic and Evolutionary Computation Conference*. New York, USA: ACM Press, pp. 695–702.

[47] H. Ishibuchi, H. Masuda, Y. Tanigaki, and Y. Nojima, "Difficulties in specifying reference points to calculate the inverted generational distance for many-objective optimization problems," in *2014 IEEE Symp. on Computational Intelligence in Multi-Criteria Decision-Making (MCDM)*, pp. 170–177.

[48] J. D. Knowles, L. Thiele, and E. Zitzler, "A Tutorial on the Performance Assessment of Stochastic Multiobjective Optimizers," *TIK-Report*, vol. 214, 2006.

[49] G. Boeing, "OSMnx: New methods for acquiring, constructing, analyzing, and visualizing complex street networks," *Computers, Environment and Urban Systems*, vol. 65, pp. 126–139, 9 2017.

[50] A. J. Nebro, J. J. Durillo, and M. Vergne, "Redesigning the jMetal Multi-Objective Optimization Framework," in *Proc. of the Companion Publication of the 2015 on Genetic and Evolutionary Computation Conference*. New York, USA: ACM Press, 2015, pp. 1093–1100.

Jens Weise received his B.Sc. degree in computer systems in engineering and his M.Sc. in medical systems engineering at the Otto von Guericke University Magdeburg in 2013 and 2017, respectively, where he is currently pursuing the Ph.D. degree with the Institute for Intelligent Cooperating Systems. His current research includes many-objective pathfinding and route planning using evolutionary multi-objective optimisation methods. He also researches methods of self-organisation in the field of factory planning.

Sanaz Mostaghim is a professor of computer science at the Otto von Guericke University Magdeburg, Germany. She holds a PhD degree in electrical engineering and computer science from the University of Paderborn, Germany. She worked as a postdoctoral fellow at ETH Zurich in Switzerland and as a lecturer at Karlsruhe Institute of technology (KIT), Germany. Her research interests are in the area of evolutionary multi-objective optimisation and decision-making, swarm intelligence, and their applications in robotics, science and industry. She is as an associate editor for the *IEEE Transactions on Evolutionary Computation* and *IEEE Transactions on Artificial Intelligence*.

APPENDIX A
SUPPLEMENTARY MATERIALS

A. Obstacles on the map

The following equations are defined for the Lake g_{LA} and Checkerboard g_{CH} obstacles. These equations are mapped to the grids with the grid cell positions of (x, y). With these functions, cells of the grid can be identified where obstacles will be positioned, hence cells with $v_{max} = 0$. The two provided obstacle functions can be used as a constraining function when running an optimisation algorithm. They take the cell's coordinate as an input and output a *True* or *False* value determining if the specified cell is an obstacle. For all obstacle functions holds: $\{x \in \mathbb{N} | 1 \leq x \leq x_{max}\}$ and $\{y \in \mathbb{N} | 1 \leq y \leq y_{max}\}$.

$$g_{CH}(x,y) = \text{sign}\left(\sin\left(\frac{\pi}{2} + \pi\, x\right)\right) + \text{sign}\left(\sin\left(\frac{\pi}{2} + \pi\, y\right)\right) - 2\,\Pi\,(x - x_{max})\,\Pi\,(y - y_{max}) = 2 \tag{8a}$$

$$\Pi(x) = H\left(x + \frac{1}{2}\right) - H\left(x - \frac{1}{2}\right) \tag{8b}$$

where $H(x)$ is the so-called Heaviside step function.

$$g_{LA}(x,y) = \left(x - 1 - \frac{x_{max}}{2}\right)^2 + \left(y - 1 - \frac{y_{max}}{2}\right)^2 - (r\, x_{max})^2 < 0 \tag{9}$$

where r denotes the radius ratio.

B. Velocity functions

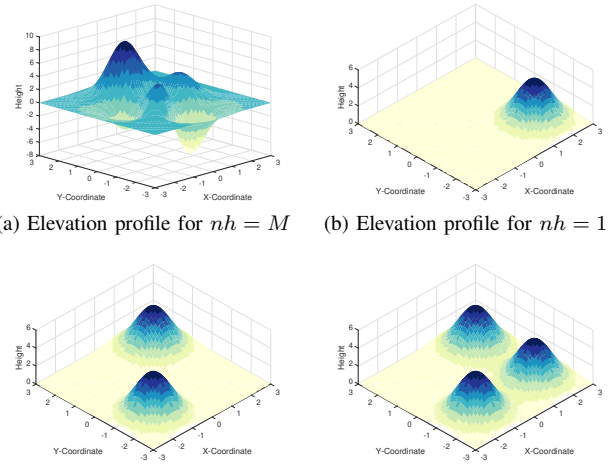
To determine the velocity v_{max} of each cell, except obstacle cells with $v_{max} = 0$, we have used the following equation, representing three street types, i.e. highways, country roads and city streets, derived from the usual speed-limits in Germany. The function takes the cell's coordinates as an input and outputs the respecting v_{max} for that cell. The provided values can be exchanged or extended to represent other road networks. For all the velocity function holds: $\{x \in \mathbb{N} | 1 \leq x \leq x_{max}\}$ and $\{y \in \mathbb{N} | 1 \leq y \leq y_{max}\}$.

$$v_{max}(x,y) = \begin{cases} 130, & \text{if } w(x,y) > 0.9 \\ 50, & \text{if } w(x,y) < -0.4 \\ 100, & \text{else} \end{cases} \tag{10}$$

where $w(x,y) = \max\left(sin(x-1), cos(y-1)\right)$.

Derived from this property, also the expected delay per path segment is defined.

$$delay(n_i, n_{i+1}) = \begin{cases} 2 & \text{if } v_{max}\,(n_i) \neq v_{max}\,(n_{i+1}) \\ 3 & \text{if } v_{max}\,(n_i) = v_{max}\,(n_{i+1}) = 50 \\ 1 & \text{if } v_{max}\,(n_i) = v_{max}\,(n_{i+1}) = 100 \\ \frac{1}{5} & \text{otherwise} \end{cases} \tag{11}$$



(a) Elevation profile for $nh = M$ (b) Elevation profile for $nh = 1$

(c) Elevation profile for $nh = 2$ (d) Elevation profile for $nh = 3$

Fig. 8. Different elevation profiles of the proposed benchmark

TABLE III
INTEGER SEQUENCES OF POSSIBLE NUMBER OF PATHS FROM THE NORTH-WESTERN CORNER TO THE SOUTH-EASTERN ONE WITH NO OBSTACLES ON OEIS.ORG FOR THE SIZE OF NXN *ASLETISMAC_NO_Xn_Yn_P_K(2,3)_B(F,T)*

Benchmark type	Integer sequence
K3,BF	A001850
K3,BT	A140518
K2,BF	A000984
K2,BT	A007764

C. Height functions

In addition to the presented height functions, we show in Figure 8 a visual representation of the different available options.

D. Number of possible paths

The number of possible paths are represented by specific integer sequences (visible at oeis.org) and is shown in Table III:

E. Data sets and Code

To enable researchers to use the proposed benchmark, we also publish the code to generate different benchmark instances and the obtained true Pareto-fronts and sets. Everything can be downloaded here: https://ci.ovgu.de/Publications/TEVC_WM_2020-p-910.html. We used Java and the jMetal framework in version 6 [50]. However, the code enables researchers to create different grids and export them as a *csv-file* to import it in other software or to use other programming languages. The codes also contain a readme file.

F. Real-World Data

OpenStreetMap provides the GPS-coordinates for a grid representation which can be easily used to measure the path length for the first objective. As for the second objective

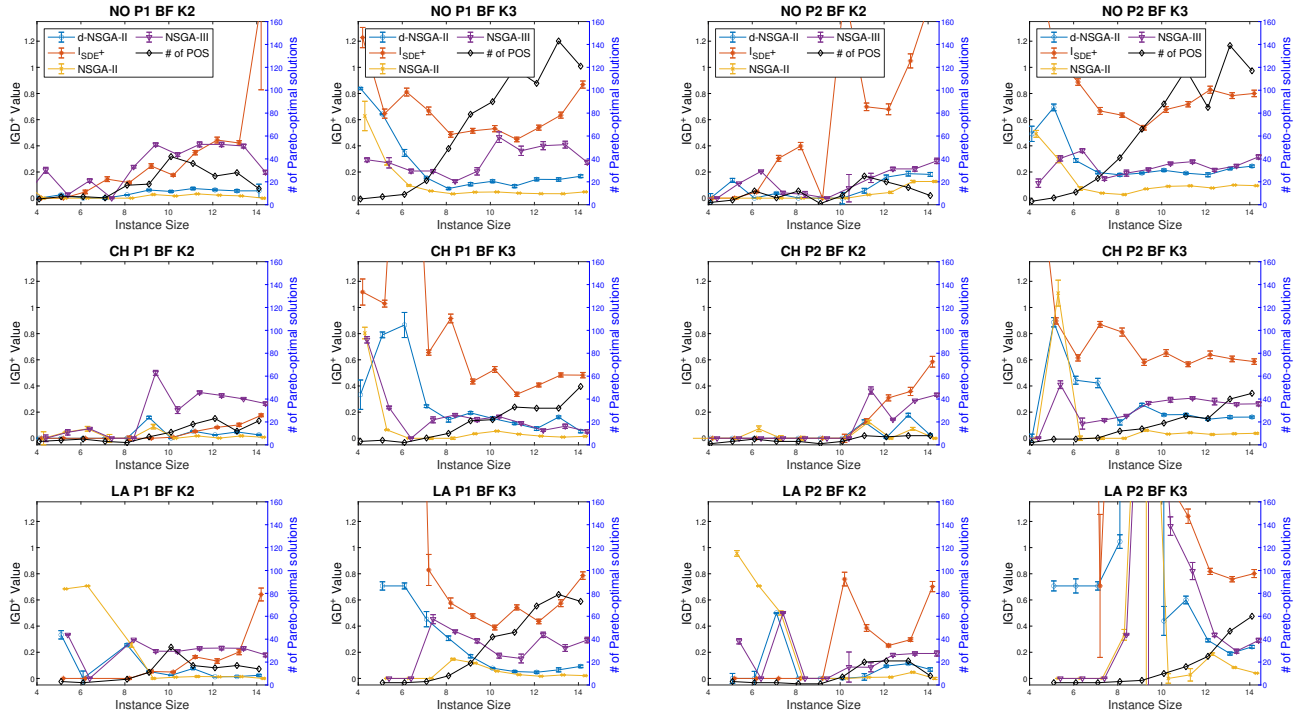
concerning the delay (number of accidents), we used the publicly available accident statistic data from 2018[3] and mapped them to the imported network. Since the coordinates of the accidents are mostly different from the available nodes in the network, we defined an R-tree index [41] on the network and performed a nearest node search for each accident. In this way, we aligned each accident to a node in the network. The third objective was measured using the Google Maps Elevation API[4]. The elevation is obtained in meters over the sea level and written to the node's properties. For the smoothness, we simplified the network to straight connections between nodes. Therefore, it is obtained in the same way as in the proposed benchmark. From the OpenStreetMap network, we could also obtain the information about speed limits per street segment. We calculated the time needed per segment as the ratio of distance and speed. Summing up the values of each segment results in the total traveling time (Objective 5). For the experiments, we take the same parameter settings as above with only one-point cross-over.

G. Results

Figure 9 illustrates the obtained IGD^+ values with respect to the different types of the problem instances. Figures 10 to 14 show true Pareto-fronts, sets and results from the algorithms for five different instances. For the smoothness objective, values are given in degrees.

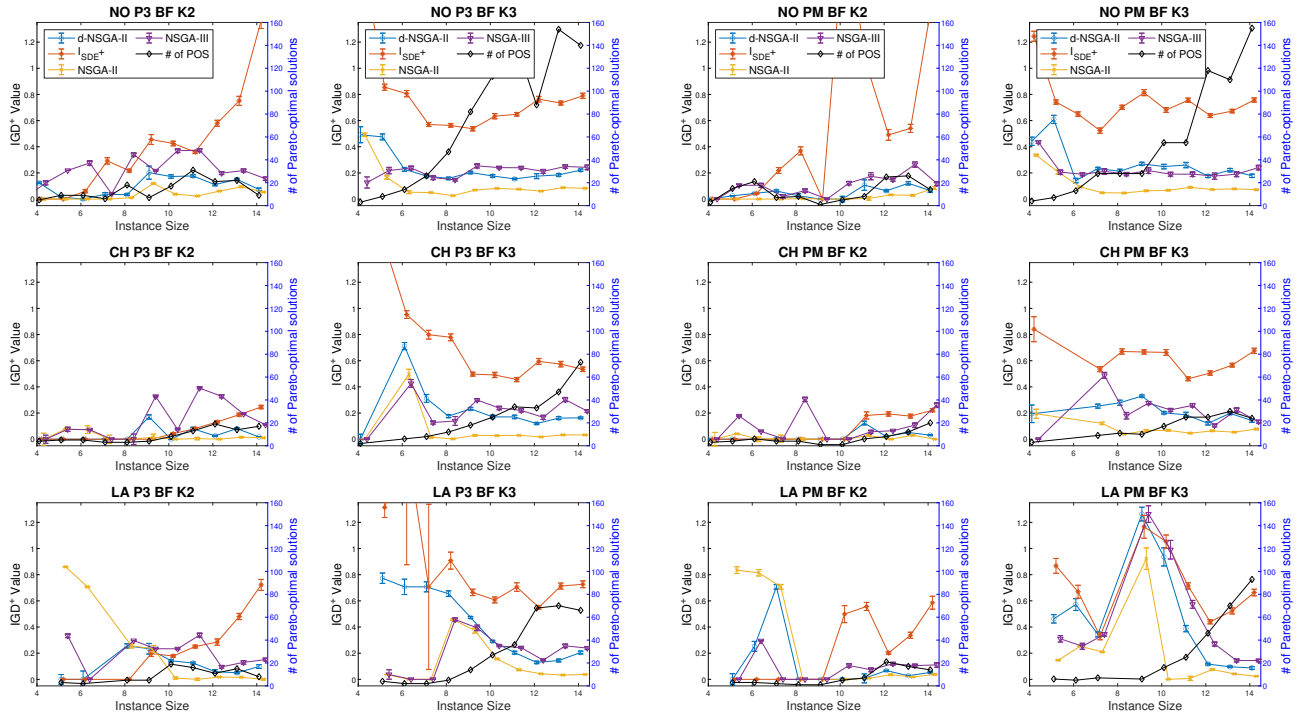
[3]https://web.archive.org/web/20200704125405/https://unfallatlas.statistikportal.de/_opendata2019.html

[4]https://developers.google.com/maps/documentation/elevation/start



(a) IGD$^+$ Values for all P1 instances

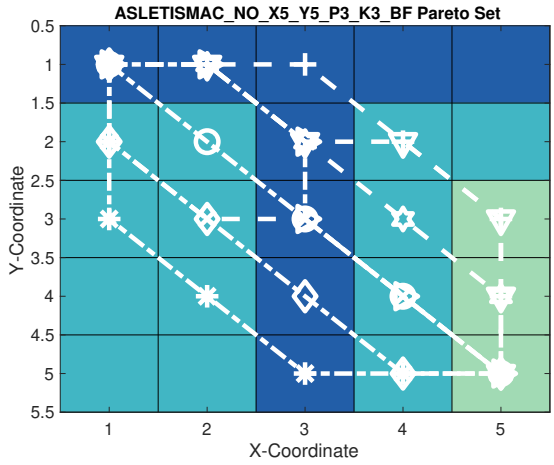
(b) IGD$^+$ Values for all P2 instances



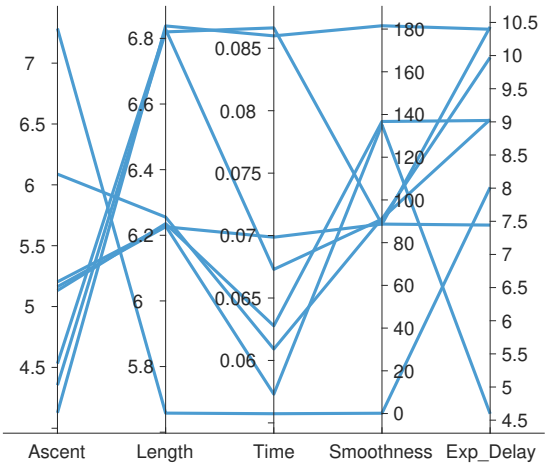
(c) IGD$^+$ Values for all P3 instances

(d) IGD$^+$ Values for all PM instances

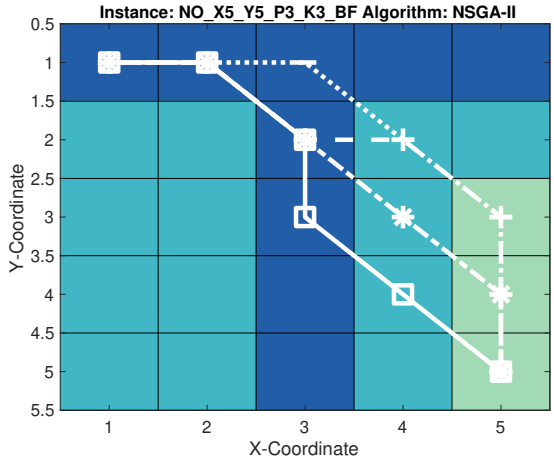
Fig. 9. The obtained IGD$^+$ values with respect to the different type, ordered by instance size



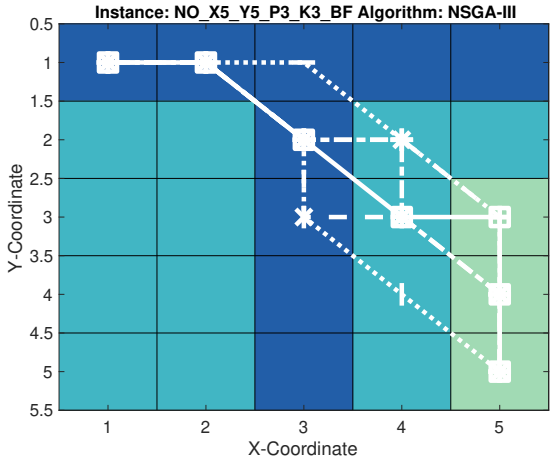
(a) Pareto-Set of instance ASLETISMAC_NO_X5_Y5_P3_K3_BF



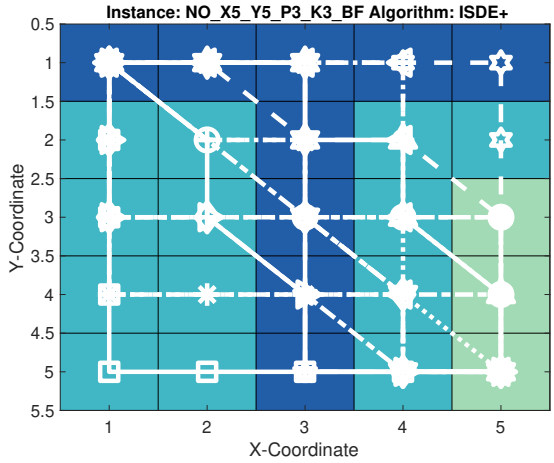
(b) Pareto-Front of instance ASLETISMAC_NO_X5_Y5_P3_K3_BF



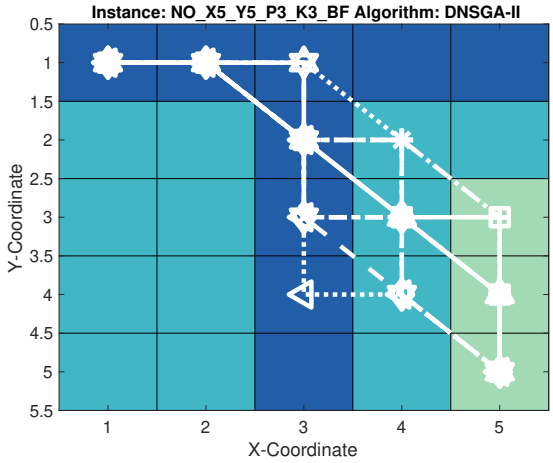
(c) Result set of algorithm NSGA-II



(d) Result set of algorithm NSGA-III

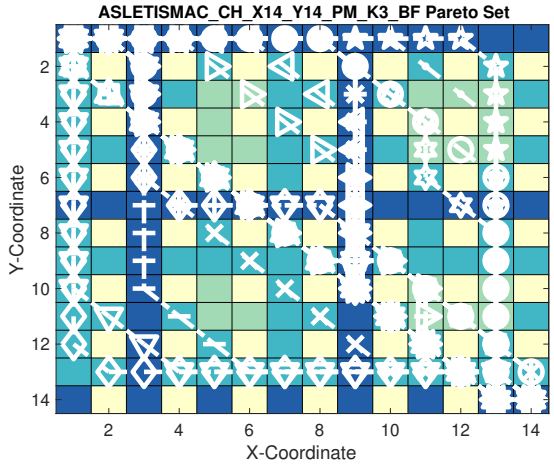


(e) Result set of algorithm $I_{SDE}+$

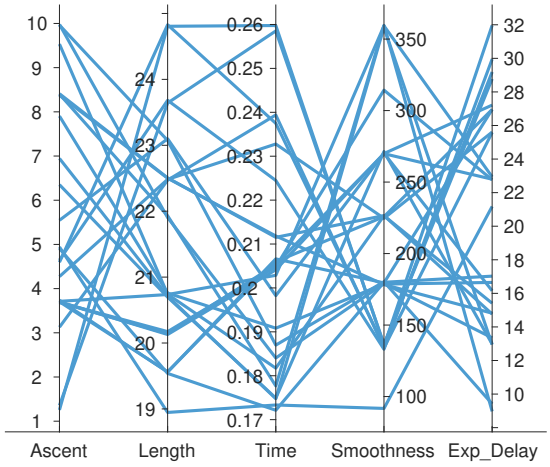


(f) Result set of algorithm D-NSGA-II

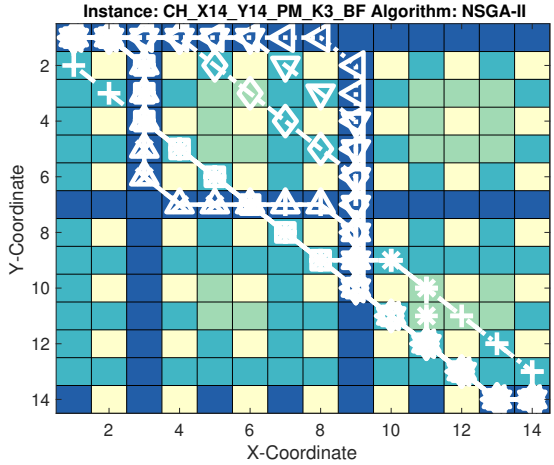
Fig. 10. Pareto-Set and Front of instance ASLETISMAC_NO_X5_Y5_P3_K3_BF and result sets of all algorithm (median run with respect to IGD^+ value)



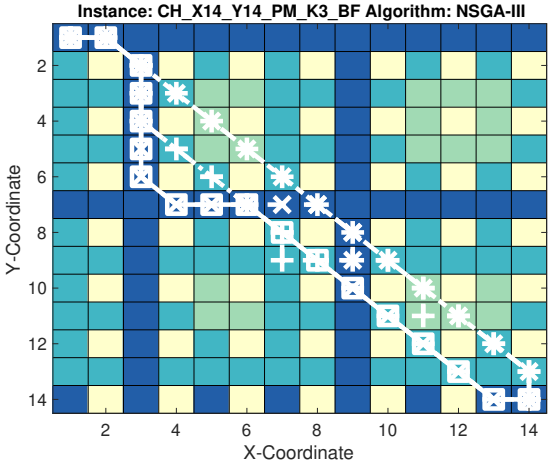
(a) Pareto-Set of instance ASLETISMAC_CH_X14_Y14_PM_K3_BF



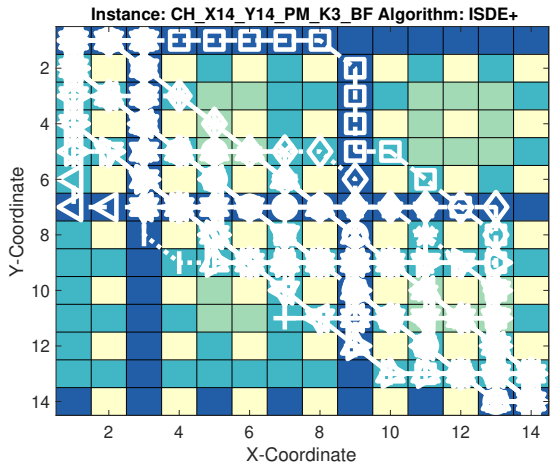
(b) Pareto-Front of instance ASLETISMAC_CH_X14_Y14_PM_K3_BF



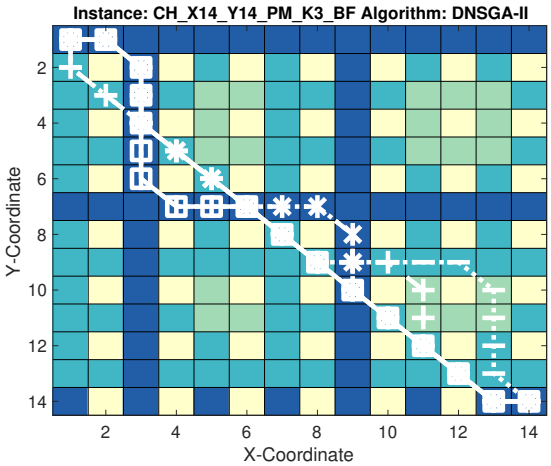
(c) Result set of algorithm NSGA-II



(d) Result set of algorithm NSGA-III

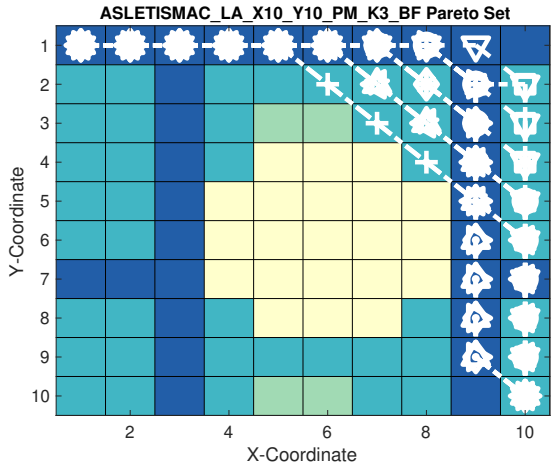


(e) Result set of algorithm I$_{SDE}$+

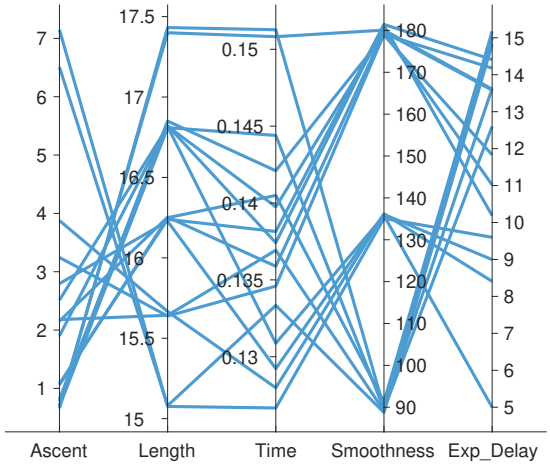


(f) Result set of algorithm D-NSGA-II

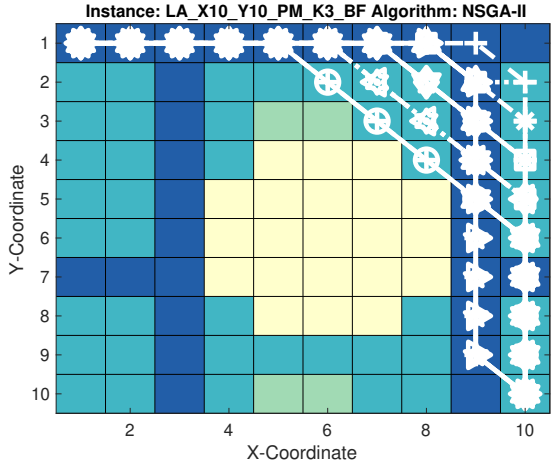
Fig. 11. Pareto-Set and Front of instance ASLETISMAC_CH_X14_Y14_PM_K3_BF and result sets of all algorithm (median run with respect to IGD$^+$ value)



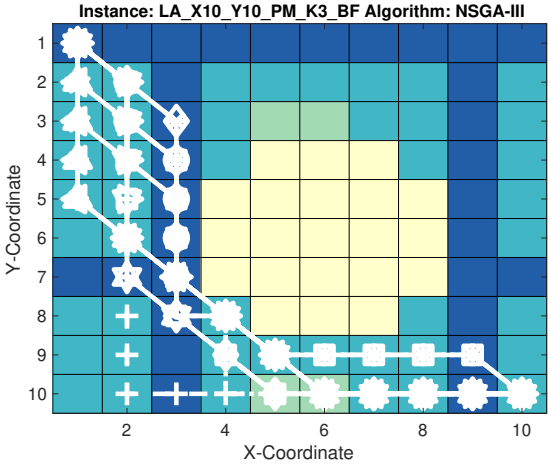
(a) Pareto-Set of instance ASLETISMAC_LA_X10_Y10_PM_K3_BF



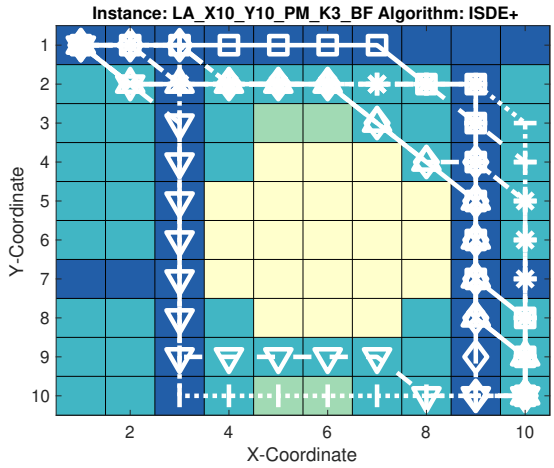
(b) Pareto-Front of instance ASLETISMAC_LA_X10_Y10_PM_K3_BF



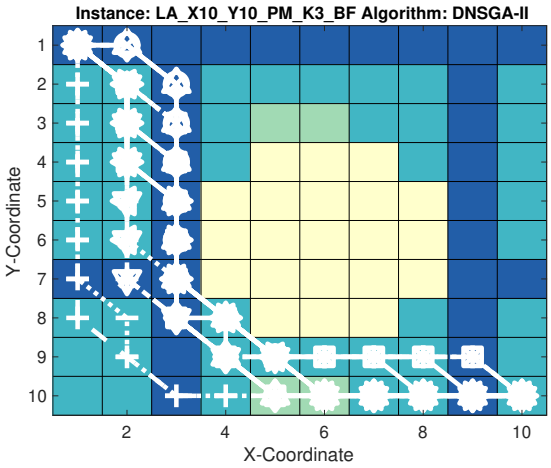
(c) Result set of algorithm NSGA-II



(d) Result set of algorithm NSGA-III

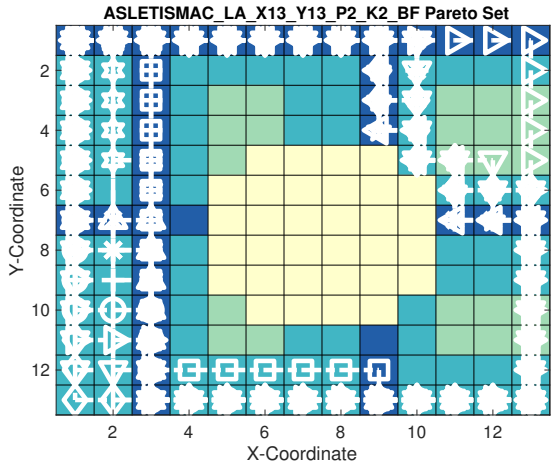


(e) Result set of algorithm I$_{SDE}$+

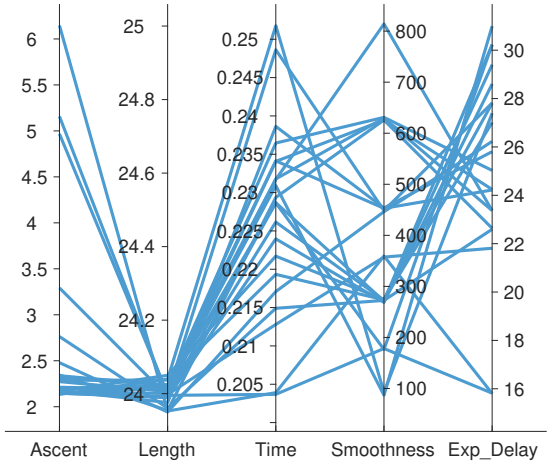


(f) Result set of algorithm D-NSGA-II

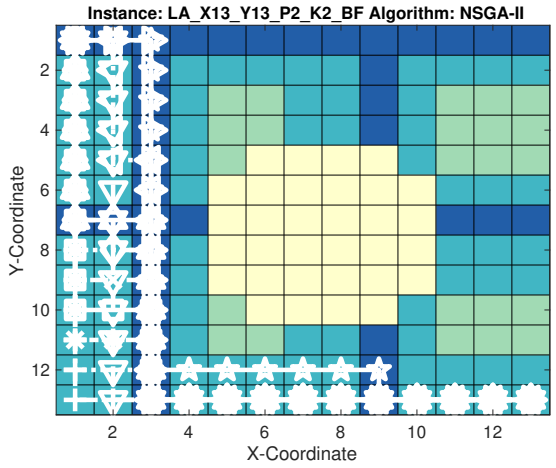
Fig. 12. Pareto-Set and Front of instance ASLETISMAC_LA_X10_Y10_PM_K3_BF and result sets of all algorithm (median run with respect to IGD$^+$ value)



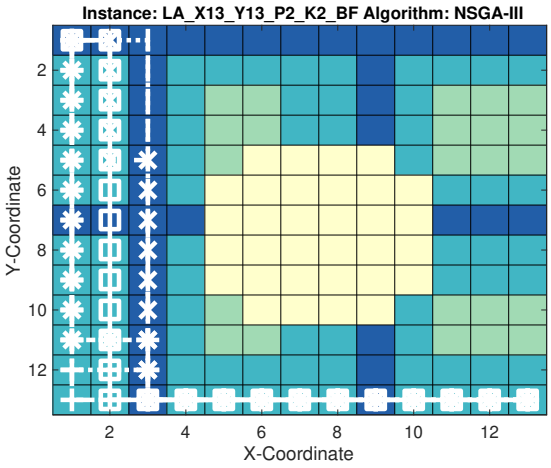
(a) Pareto-Set of instance ASLETISMAC_LA_X13_Y13_P2_K2_BF



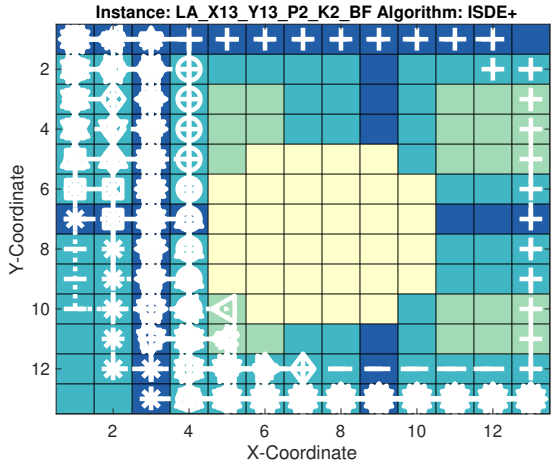
(b) Pareto-Front of instance ASLETISMAC_LA_X13_Y13_P2_K2_BF



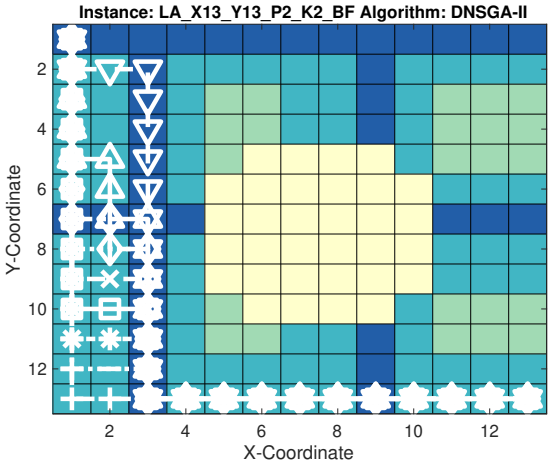
(c) Result set of algorithm NSGA-II



(d) Result set of algorithm NSGA-III

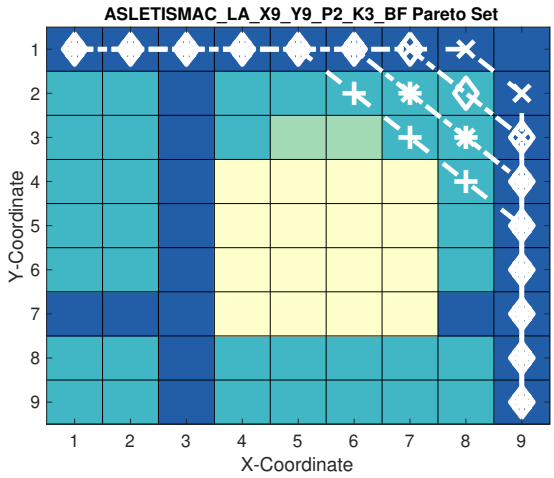


(e) Result set of algorithm $I_{SDE}+$

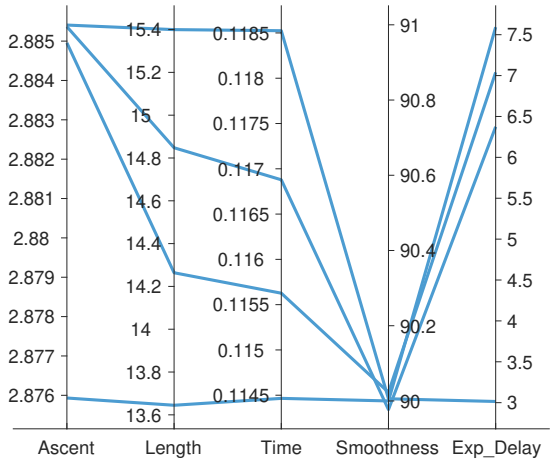


(f) Result set of algorithm D-NSGA-II

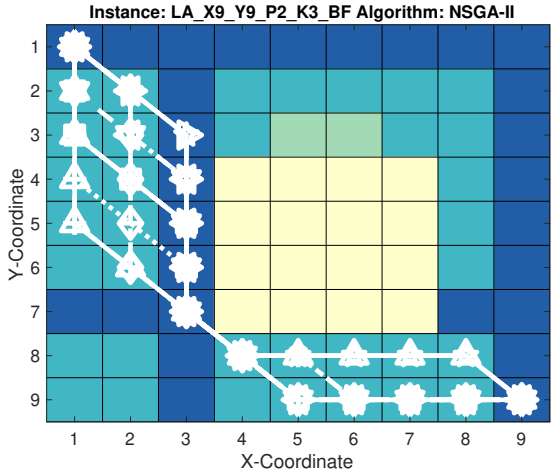
Fig. 13. Pareto-Set and Front of instance ASLETISMAC_LA_X13_Y13_P2_K2_BF and result sets of all algorithm (median run with respect to IGD^+ value)



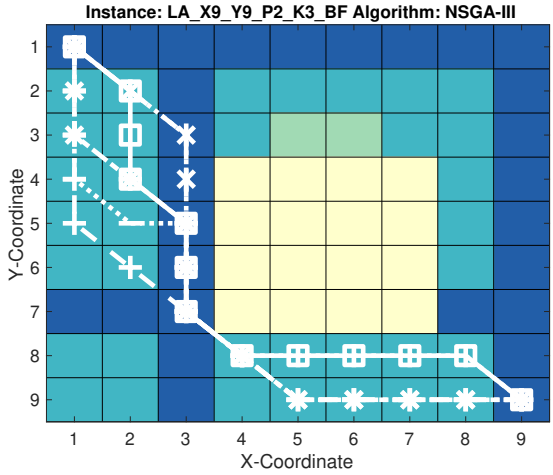
(a) Pareto-Set of instance ASLETISMAC_LA_X9_Y9_P2_K3_BF



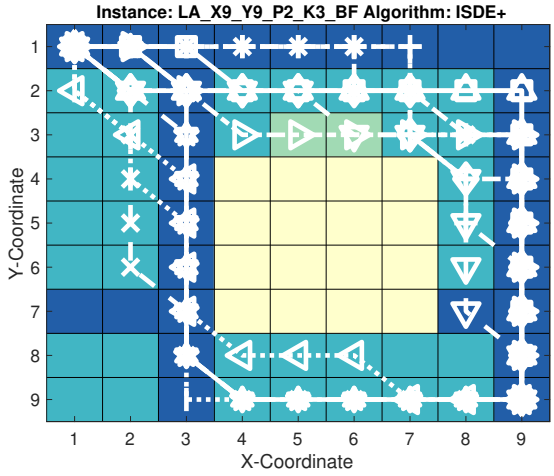
(b) Pareto-Front of instance ASLETISMAC_LA_X9_Y9_P2_K3_BF



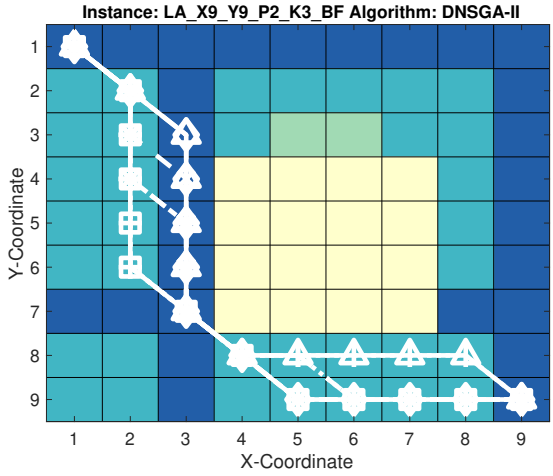
(c) Result set of algorithm NSGA-II



(d) Result set of algorithm NSGA-III



(e) Result set of algorithm $I_{SDE}+$



(f) Result set of algorithm D-NSGA-II

Fig. 14. Pareto-Set and Front of instance ASLETISMAC_LA_X9_Y9_P2_K3_BF and result sets of all algorithm (median run with respect to IGD^+ value)