Hybrid Evolutionary Approach to Multi-objective Path Planning for UAVs

Nikolas Hohmann*, Mariusz Bujny†, Jürgen Adamy* and Markus Olhofer†
*Control Methods & Robotics Lab, Technical University of Darmstadt, Darmstadt, Germany
Email: {nikolas.hohmann, adamy}@rmr.tu-darmstadt.de
†Honda Research Institute Europe GmbH, Offenbach, Germany
Email: {mariusz.bujny, markus.olhofer}@honda-ri.de

***Abstract*—The goal of Multi-Objective Path Planning (MOPP) is to find Pareto-optimal paths for autonomous agents with respect to several optimization goals like minimizing risk, path length, travel time, or energy consumption. In this work, we formulate a MOPP for Unmanned Aerial Vehicles (UAVs). We utilize a path representation based on Non-Uniform Rational B-Splines (NURBS) and propose a hybrid evolutionary approach combining an Evolution Strategy (ES) with the exact Dijkstra algorithm. Moreover, we compare our approach in a statistical analysis to state-of-the-art exact (Dijkstra's algorithm), gradient-based (L-BFGS-B), and evolutionary (NSGA-II) algorithms with respect to calculation time and quality features of the obtained Pareto fronts indicating convergence and diversity of the solutions. We evaluate the methods on a realistic 2D urban path planning scenario based on real-world data exported from OpenStreetMap. The examination's results indicate that our approach is able to find significantly better solutions for the formulated problem than standard Evolutionary Algorithms (EAs). Moreover, the proposed method is able to obtain more diverse sets of trade-off solutions for different objectives than the standard exact approaches. Thus, the method combines the strengths of both approaches.**

***Index Terms*—multi-objective optimization, path planning, hybrid algorithms, evolutionary algorithms, UAV, unmanned aerial vehicle**

I. Introduction

Single-objective path planning problems for mobile robotic applications have been well-studied for the last decades [1]. These approaches often account for finding a shortest path for a single robot known to be the only agent in a cluttered environment. A rising interest in Multi-Objective Path Planning (MOPP) approaches has been developing with the increasing integration of autonomous agents in real-world applications, where different robots and humans act in the same environment. Such multi-agent, human-machine systems result in much more complex path planning problems. Not only Euclidean distances or obstacles but also other objectives need to be considered concurrently. Possible objectives can be differentiated into two groups related to: 1) robotic design requirements, and 2) demands of different stakeholders in the robot's environment. Examples for the former category are minimizing energy consumption, satisfying actuator saturations, or maximizing the quality of communication signals. Examples for the latter category are minimizing the risk of harming a human, minimizing the noise immission on humans, or avoiding the intersection with other agent's paths.

In lots of cases, optimizing different objectives independently leads to paths of different shapes. That is why it is important to be able to provide some trade-off solutions, which are generally not optimal with respect to only one objective, but allow for finding a good compromise among all objectives. Those solutions build up the so-called Pareto-optimal front in the objective space. Finding the Pareto-optimal front is the aim of Multi-Objective Optimization (MOO). In this work, we focus on developing MOO techniques with emphasis on path planning.

In Section I-A we will present and categorize different classes of optimizers that solve the MOPP problem. Conventional path planning techniques rely on gradient-based or exact optimizers. They are fast and nearly or, under some assumptions [2], completely optimal in solving single-objective optimization problems. But, they show drawbacks in the optimization of multiple objectives or multimodal problems. In recent years, especially meta-heuristic path planning approaches, like Evolutionary Algorithms (EAs), have spread [3]. EAs have been shown to perform very well on MOO problems, identifying a well-diversified Pareto set also for multimodal problems. But, EAs have disadvantages as they usually need more computational resources and are not able to guarantee optimality. In Section II we propose a new hybrid EA that combines the benefits of both classes of optimizers.

Motivated by Tovey [4] we have the ambition to benchmark our approach not only exclusively within the research area of nature-inspired algorithms, but also within the area of exact and gradient-based solvers. This is why we compare our approach in Section III to the state-of-the-art multi-objective genetic algorithm NSGA-II (Non-dominated Sorting Genetic Algorithm) [5] as well as to the gradient-based optimizer L-BFGS-B (Limited-memory Broyden-Fletcher-Goldfarb-Shanno algorithm with Bounds) [6] and to the graph-based Dijkstra algorithm [7]. We evaluate the performances of the different approaches on the formulated MOPP problem on a real-world UAV path planning scenario in the city of Darmstadt, Germany. To the best of our knowledge, we are the first to conduct examinations on the MOPP problem with such a versatile group of benchmark solvers.

We show that our hybrid approach performs significantly better than contemporary EAs while avoiding the drawbacks of exact and gradient-based solvers. Finally, we conclude the paper and give an outlook to interesting future investigations in Section IV.

A. Related Work

Autonomous robots are used in all kinds of environments, e.g., underwater, on the water, on the ground, or in the air. Thus, the range of objectives for MOPP is very wide. We cluster MOPP approaches according to different categories that are: 1) the types of objectives, 2) the way how multiple objectives are handled, 3) the used optimizers, and 4) the representation schemes.

1) Types of objectives: In many approaches the length of the path is used as one objective [8]–[19]. Babel [13] and Yin [20] refine this objective to the travel time of the agent. Oberge et al. [12], Ganganath et al. [15], and Ma et al. [18] derive different energy consumption models and compute the energy needed to travel along a certain path in order to minimize it. A group of path planners [8]–[12], [16]–[18], [20] aims to maximize the clearance to static obstacles or defined danger zones. Other approaches [9], [12], [13], [16]–[18] consider various feasibility objectives like a path's smoothness, minimal curvature, or its maximum altitude. Furthermore, some works take a safety objective into account, which can be seen from two different perspectives: Ahmed et al. [11] ensure safety through maximization of the robot's range of vision in its environment, whereas Babel's [13] approach minimizes the agent's chance to be detected by other agents. Lastly, Mittal et al. [14] and Rubio et al. [19] introduce some risk metrics that quantify the damage resulting from a potential crash for an agent following a certain path.

2) Handling of multiple objectives: A group of approaches [12], [17], [20] follows a common way to handle multiple objectives with a single-objective optimizer by a weighted aggregation of the objectives into a single objective function. Babel [13] is choosing one of the several proposed objectives to optimize it separately and thus can not provide any Pareto front. Jalel et al. [16] are applying several single-objective optimization stages, which has the effect that solutions of a latter stage can get worse with respect to objectives of a previous stage. The other approaches [8]–[11], [14], [15], [19], [20] handle two objectives in a multi-objective manner. They differ from our approach in terms of the used objectives like stated in 1), the utilized solver and the developed representation scheme, whose consideration follows in 3) and 4).

3) Used solvers: Some works use different graph-based, single-objective solvers to solve their path planning problems, respectively. Jalel et al. [16] utilize a version of the Bellman-Ford algorithm [21]. Babel [13] uses the Dijkstra algorithm [7]. Yin et al. [20] choose the A* algorithm [22], which is a generalization of the Dijkstra algorithm. Moreover, Oberge et al. [12] and Jalel et al. [17] use a single-objective genetic algorithm (GA). Finally, several optimizers are able to directly handle multiple objectives. Many approaches [8], [9], [11], [14], [19] use the multi-objective genetic algorithm NSGA-II [5]. Particle swarm optimization approaches are used by Zhang et al. [10] and Ma et al. [18]. Lastly, Ganganath et al. [15] utilize a New Approach to Multi-Objective A* (NAMOA*) [23] to solve the MOPP problem.

4) Representation schemes: A path's representation that is used by a graph-based solver can only consist of a discrete set of nodes and edges. Therefore, a group of approaches [8], [9], [15], [20] uses paths that appear to be two-dimensional, grid-based, zig-zag-shaped polylines with a varying number of nodes. Similarly, Zhang et al. [10] and Ma et al. [18] choose two-dimensional straight line segments as paths. However, the representation is restricted to a fixed number of nodes. The same representation is extended by Oberge et al. [12] and Rubio et al. [19] to three dimensions. Other approaches avoid paths with sharp turns by using smooth 2D B-spline curves [11], [14] or their generalization to rational B-splines [16], [17]. Babel [13] is constructing complex paths by optimally selecting polynomial path segments from a precomputed set of segments.

B. Fundamentals

1) Multi-objective Optimization: The goal in MOO in its most general formulation is to find D-dimensional solution vectors $\mathbf{z} = \begin{bmatrix} z_1 & \dots & z_D \end{bmatrix}^T$ that minimize or maximize a set $\mathcal{F}$ of E objective functions

$$f_e(\mathbf{z}), \quad e = 1, \dots, E \tag{1}$$

that are subject to F inequality constraints

$$g_f(\mathbf{z}) \geq 0, \quad f = 1, \dots, F, \tag{2}$$

as well as to G equality constraints

$$h_g(\mathbf{z}) = 0, \quad g = 1, \dots, G. \tag{3}$$

Besides, each component of $\mathbf{z}$ can be constrained by a lower and an upper bound

$$z_d^{(L)} \leq z_d \leq z_d^{(U)}, \quad d = 1, \dots, D. \tag{4}$$

The search space $\mathcal{S}$ contains all possible solutions of the optimization problem. A solution is feasible if it satisfies all constraints and variable bounds. The set of all feasible solutions is called a feasible region. Through objective functions, a point in the search space $\mathcal{S}$ is mapped into the objective space $\mathcal{O}$. In theory, a solution that optimizes all objectives independently is called the utopia point $\mathbf{z}_{\text{Utopia}}$. De facto, it does not exist. Therefore, the concept of dominance is introduced. A solution $\mathbf{z}_1$ is said to dominate another solution $\mathbf{z}_2$ ($\mathbf{z}_1 \preccurlyeq \mathbf{z}_2$) if $\mathbf{z}_1$ is not worse than $\mathbf{z}_2$ for all objectives and $\mathbf{z}_1$ is strictly better than $\mathbf{z}_2$ in at least one objective. Every $\mathbf{z}$ in a set of solutions that is not dominated by any other solution in this set belongs to the set of non-dominated solutions. By mapping the non-dominated set into the objective space, a set of values called Pareto set is obtained.

2) Non-Uniform Rational B-Splines (NURBS): Referring to Piegl et al. [24], a NURBS curve of order $p+1$ is defined by

$$\mathbf{C}(u) = \frac{\sum_{i=0}^{n_p-1} N_{i,p}(u) w_i \mathbf{P}_i}{\sum_{i=0}^{n_p-1} N_{i,p}(u) w_i}, \tag{5}$$

where

- n_p is the number of control points,
- p is the degree of the basis function $N_{i,p}$,
- $\mathbf{P}_i = \begin{bmatrix} x_i & y_i \end{bmatrix}^T$ is the i^{th} control point (assuming a 2D curve) and
- w_i is its weight.

The basis functions $N_{i,p}$ are defined along the parameter u with respect to a defined knot vector

$$\mathbf{U} = \begin{bmatrix} u_0 & \dots & u_m \end{bmatrix}^T,$$

containing $m+1$ knots, whereas $m = n_p + p$. The basis functions can be calculated recursively using the De-Boor-Cox formulas [25], [26], [27].

II. Hybrid evolution strategy (HES)

We begin by formulating the objective functions of the tackled optimization problem in Section II-A. The definition of the optimization vector $\mathbf{z}$, thus the representation of the path, is introduced in Section II-B. After that, we propose the hybrid algorithm in Section II-C.

A. Problem definition

We consider a path planning scenario on a rectangular, two-dimensional map, which we define by $\mathbb{D} = [x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}]$. The objectives are computed based on E two-dimensional scalar fields $F_e : (x,y) \in \mathbb{D} \to \mathbb{R}$. Given a start vector $\mathbf{x}_{\text{s}} = \mathbf{C}(a) \in \mathbb{D}$ and a goal vector $\mathbf{x}_{\text{g}} = \mathbf{C}(b) \in \mathbb{D}$ the tackled MOPP problem targets finding curves $\mathcal{C} = \{\mathbf{C}(u) : u \in [a,b]\}$, so that the set of E objectives $\mathcal{F} = \{f_1, \dots, f_E\}$ is minimized. All objectives are defined by the line integral

$$f_e(\mathbf{z}) = \int_{\mathcal{C}} F_e(\mathbf{C}(u)) ds \overset{!}{=} \int_a^b F_e(\mathbf{C}(u)) |\mathbf{C}'(u)| du. \tag{6}$$

The definition of the vector of optimization variables $\mathbf{z}$ will be given below.

B. Representation

Piegl et al. [24] give a detailed look into the definition of NURBS curves and their properties. When NURBS curves are used as a path representation in path planning problems, especially three properties are useful, namely,

- the *convex hull property*, meaning that the curve lies within the convex hull that is spanned by the control points,
- *local approximation*, meaning that a slight change in a control point's position will affect the curve's shape only locally around this control point, and
- *infinite differentiability* apart from knots, where the curve is $p-c$ times differentiable, with c being the multiplicity of the knot.

The former two properties are helpful in the optimization process itself, as the curve can be constrained to a bounded area and escape local minima by varying only one component of the optimization variable vector. The last property ensures the smoothness of the path on the representation level. This is particularly advantageous in case of agents that rely on smooth paths due to the limitations of the actuation systems they use. In this sense, smooth paths would thus result in a lower energy consumption.

The vector of design variables is defined as

$$\mathbf{z} = \begin{bmatrix} w_0 \; x_1 \; y_1 \; w_1 \; \dots \; x_{n_{p-1}} \; y_{n_{p-1}} \; w_{n_{p-1}} \; w_{n_p} \end{bmatrix}^T, \tag{7}$$

where the number of control points n_p is a hyperparameter of the optimization algorithm. In all experiments we choose $n_p = 15$, which was fitted empirically to the scenario size. Note that $\begin{bmatrix} x_0 & y_0 \end{bmatrix}^T = \mathbf{x}_{\text{s}}$ and $\begin{bmatrix} x_{n_p} & y_{n_p} \end{bmatrix}^T = \mathbf{x}_{\text{g}}$ are part of the problem definition and, therefore, are not subject to optimization. Thus, referring to Section I-B1, we have $D = 3n_p - 4$.

With the basis function degree $p = 2$, the knot vector

$$\mathbf{U} = \begin{bmatrix} 0 & 0 & \dots & \frac{1}{n_p - p} k & \dots & 1 & 1 \end{bmatrix}^T,$$

where $k = 0, \dots, n_p - p$ is defined such that the curve is clamped to the first and last control point.

Moreover, during the optimization, the control point positions are bounded by the size of the design domain $x_{\min} \leq x_d \leq x_{\max}$ and $y_{\min} \leq y_d \leq y_{\max}$.

C. Algorithm

There are many optimizers like Particle Swarm Optimization (PSO), Differential Evolution (DE), or Simulated Annealing (SA) that can address arbitrary quantifiable objectives and constraints. We put emphasis on a straightforward implementation as well as interpretable results. Therefore, the algorithm proposed by us is based on a standard Evolution Strategy (ES), which is known to perform well on continuous optimization problems. Later, we also compare it to a Genetic Algorithm (GA), which is a commonly used alternative. An ES optimizes so-called individuals $\mathbf{I} = \{\mathbf{z}, \boldsymbol{\sigma}\}$, which consist of the vector of design variables $\mathbf{z}$ and a vector of step sizes $\boldsymbol{\sigma} = \begin{bmatrix} \sigma_1 & \dots & \sigma_D \end{bmatrix}$. The ES is running for T generations. At generation $t = 0$, with $t \in [0, T]$, the algorithm starts by initializing the population consisting of μ parent individuals $\mathbf{I}_s$ with $s = 1, \dots, \mu$. Thereafter, an evolutionary process is carried out until a stop criterion is met. This process consists of mutation and recombination of λ offspring individuals, evaluation of those, and selection of μ individuals for the next generation. In contrast to a GA, in an ES, the step sizes σ_d of the mutation are also subject to the evolutionary process.

1) Initialization: We present the default and an advanced approach here. In the ES's default implementation, μ individuals are drawn from the search space according to the uniform random distribution. With this method, due to the representation based on control points, used in this work, the initially obtained curves would be highly suboptimal. That is why in our default implementation control points are sampled equidistantly on a straight line segment between $\mathbf{x}_{\mathrm{s}}$ and $\mathbf{x}_{\mathrm{g}}$, and are varied componentwise with Gaussian noise $\mathcal{N}(0, \sigma_{\mathrm{G}}^2)$, with a hyperparameter that we set to $\sigma_{\mathrm{G}}^2 = 5\mathrm{m}$, which was found to give good results in a prior empirical evaluation. Weights are initialized with $w_i = 1$, whereas the initial step size vector $\boldsymbol{\sigma}_0$ is set by the user. With this initialization method, the ES takes a long time to converge to a near-optimal path if the optimal solution differs greatly in shape from a straight line. This is why we propose a hybrid ES that includes an advanced initialization phase for the control point positions. In a preprocessing step any user-defined single-objective optimizer can be used to find n_{NOS} near-optimal solutions for the derived weighted and aggregated single-objective optimization function

$$f_{\mathrm{A}}(\mathbf{z}) = \rho_1 f_1(\mathbf{z}) + \ldots + \rho_E f_E(\mathbf{z}), \tag{8}$$

where $\sum_{e=1}^{E} \rho_e = 1$ holds. The obtained solutions are used to initialize n_{NOS} individuals. The remaining $\mu - n_{\mathrm{NOS}}$ parent individuals are initialized according to the default approach.

2) Variation: From μ parent individuals, the ES calculates λ offspring individuals by randomly selecting a parent individual for λ times and applying variation operators to it. We use two different variation operators. Firstly, a step size crossover that averages the step sizes of two randomly selected individuals $\mathbf{I}_{i,t}$ and $\mathbf{I}_{j,t}$ at generation t yields

$$\boldsymbol{\sigma}_{i,t+1} = \boldsymbol{\sigma}_{j,t+1} = 0.5\left(\boldsymbol{\sigma}_{i,t} + \boldsymbol{\sigma}_{j,t}\right). \tag{9}$$

Secondly, a mutation operator based on the extended log-normal rule [28] is applied to every individual $\mathbf{I}_{s,t}$. For this purpose its step size vector is adapted initially, yielding

$$\boldsymbol{\sigma}_{s,t+1} = e^{\tau_0 \xi_0} \begin{bmatrix} \sigma_{1,t} e^{\tau \xi_1} & \ldots & \sigma_{D,t} e^{\tau \xi_D} \end{bmatrix}^T. \tag{10}$$

Here, all $\xi_d \sim \mathcal{N}(0,1)$ are different random numbers drawn from the standard normal distribution. Furthermore, we use the default values of $\tau_0 = \frac{l}{\sqrt{2D}}$ as well as $\tau = \frac{l}{\sqrt{2\sqrt{D}}}$, with l, being a hyperparameter that is set to $l = 1$. Then the individual itself is adapted with the newly calculated step sizes

$$\mathbf{I}_{r,t+1} = \begin{bmatrix} w_{0,t} + \sigma_{1,t+1}\xi_1 & \ldots & w_{n_p,t} + \sigma_{D,t+1}\xi_D \end{bmatrix}^T, \tag{11}$$

where $r = 1, \ldots, \lambda$ and $\xi_d \sim \mathcal{N}(0,1)$ holds.

3) Selection: In order to select μ new individuals from the λ varied children, we chose a selection operator based on the number of objectives E. For $E = 2$ the non-dominated and crowding distance sorting selection operator from NSGA-II [5] has proven to work well. For $E \geq 3$ the selection operator from NSGA-III [29] as well as the selection operator from RVEA (Reference Vector guided Evolutionary Algorithm) [30] have been tested. The former works with non-dominated sorting and a nearest-to-reference-point selection, while the latter is matching individuals to niches by choosing minimal angles to defined reference vectors and then selecting them based on an angle-penalized distance measure.

4) Stop criterion: To stop the optimization process, the best achieved fitness value is tracked with regard to all E objectives. When none of them has improved for t_T generations, where t_T is a user-defined parameter, the optimization terminates.

III. Evaluation

In this section, we test the proposed algorithm on a real-world scenario, which is introduced in Section III-A. We evaluate it against state-of-the-art solvers presented in Section III-B. Parameters for the conducted experiments are given in Section III-C. We will introduce different metrics that are used for the comparison in Section III-D before demonstrating the results of the experiments in Section III-E.

A. Scenario

The evaluation is based on OpenStreetMap (OSM) [31] data of a map section that is visualized in Fig. 1 and shows the German city of Darmstadt. We use the OSM data to sample different low-level grid maps, representing, e.g., the city's road network, or the height of buildings. The grid maps are discretized according to resolutions x_{res} and y_{res} in x and y direction, respectively. We use these low-level grid maps to derive exemplary high-level grid maps:

- For a flying agent, we assume that a crash landing on a roof will cause less damage to humans than a crash landing anywhere else in the city. Therefore, we suppose paths that go over buildings to have a lower risk of harming a human than paths over other areas. Based on the buildings map, the derived risk map in Fig. 2 can be calculated such that for every grid cell applies: the further away the nearest building, the higher the risk.
- In addition to this, we also want to model a noise immission property. We assume that paths that go over streets will be perceived as less noisy by humans compared to paths that go over other areas. Therefore, we introduce a noise map that can be seen in Fig. 3. The noise value in every cell increases linearly with the distance to the nearest road.

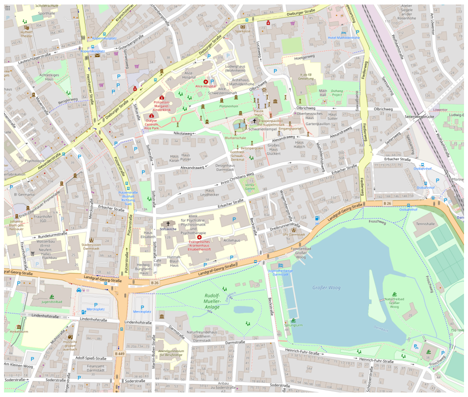
Fig. 1: Map of the city of Darmstadt, Germany, imported from OSM and used for the evaluation.

The described high-level grid maps are used in the optimization, replacing the scalar fields F_e in the line integral (6). Please note that due to the discretization, the continuous

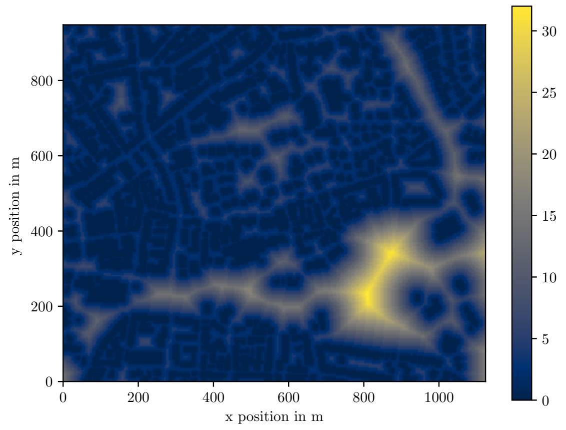

Fig. 2: Risk map that was generated with underlying OSM data. In this risk model, grid cells with lighter colors indicate a higher risk, i.e., greater damage, in case of a failure if the particular grid cell belongs to the path.

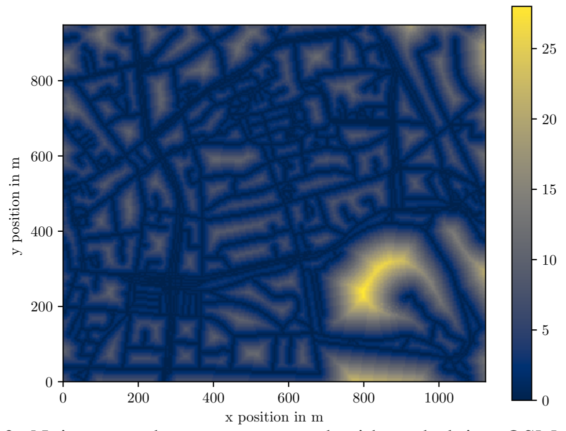

Fig. 3: Noise map that was generated with underlying OSM data. In this noise model, grid cells with lighter colors indicate a higher noise immission on humans if a planned path goes through the particular grid cell.

integral is estimated by a trapezoidal rule integration. The integral is calculated on the high-level grid-maps, which are interpolated by a bivariate spline surface interpolation.

B. Benchmark solvers

We compare our approach with the state-of-the-art multi-objective optimizer NSGA-II [5], which uses polynomial mutation and simulated binary crossover as variation operators. Compared to our approach, the same selection operator is used for bi-objective problems, but NSGA-II lacks the evolutionary step size control.

Furthermore, we also solve the problem with the gradient-based optimizer L-BFGS-B [6], a variant of the widely-used L-BFGS algorithm [32], which is capable of incorporating variable bounds. Moreover, our approach is compared with a bidirectional version of the famous Dijkstra algorithm [7], which finds shortest paths in graphs.

Please note that the NSGA-II and L-BFGS-B approaches also use the control-point-based representation (7) of our method. However, this representation can not be adapted to the Dijkstra algorithm because Dijkstra's method has a graph-based search space and, therefore, calculates a graph-based solution. In order to fit the Dijkstra algorithm to our problem, the grid map is transformed into a weighted graph based on the cell entries of the grid map. The Dijkstra algorithm will then find the optimal zig-zag path from $\mathbf{x}_\mathrm{s}$ to $\mathbf{x}_\mathrm{g}$. Eventually, to compare the output with the other solver's solutions, the derived path is approximated by a NURBS curve using the least square method with the same number of control points n_p as in the other approaches.

We want to clarify that Dijkstra and L-BFGS-B are not capable of handling more than one objective at once. Therefore, to maintain comparability, we weight and aggregate multiple objectives into a single one as also done in equation (8). A Pareto front can then be obtained by running $n_{\mathrm{MO,sweep}}$ optimizations for different values of the weights ρ_e.

C. Parameters

In Table I, we give an overview of the most important parameters for each approach. Parameters that were not introduced by us are set to standard values from literature. Please note that in the following, our hybrid ES with advanced initialization is abbreviated as HES, while the version with the default initialization is called ES. In the same way, NSGA-II with default and hybrid initialization are referred to as NS and HNS, respectively. As abbreviation for L-BFGS-B we introduce LB, for the Dijkstra algorithm DIJ, and for the Dijkstra with NURBS curve approximation ADIJ. We compare the solvers for a broad set of different start and end positions $\mathbf{x}_\mathrm{s}$ and $\mathbf{x}_\mathrm{g}$. To obtain a representative set for the statistical analysis, we randomly generate 10^7 start and end point combinations and weight them according to

TABLE I: Setup of the optimization methods used in the paper

	Parameter	**Symbol**	**Value**
General	scenario dimensions	$[x_{\min}\ x_{\max}]$	$[0 \quad 1124]$
		$[y_{\min}\ y_{\max}]$	$[0 \quad 948]$
	# of objectives	E	2
	discretization resolution	$x_{\mathrm{res}}, y_{\mathrm{res}}$	4m, 4m
	# of control points	n_p	15
	basis function degree	p	2
Method			
ES	initial step size	$[\sigma_{x,0}\ \sigma_{y,0}\ \sigma_{w,0}]$	$[1\ \ 1\ \ 0]$
	parent population size	μ	15
	offspring population size	λ	100
	generation threshold	t_T	10
HES	# of pre-calculations	n_{NOS}	5
	preprocessing optimizer		ADIJ
NS	crowding degree	$\eta_{\mathrm{crossover}}$	20
	crossover probability	$p_{\mathrm{crossover}}$	0.9
	crowding degree	η_{mutation}	20
	mutation probability	p_{mutation}	$1/D$
	population size	N	100
	generation threshold	t_T	10
HNS	# of pre-calculations	n_{NOS}	5
	preprocessing optimizer		ADIJ
LB DIJ ADIJ	# of optimizations	$n_{\mathrm{MO,sweep}}$	65

$$w(\mathbf{x}_\mathrm{s}, \mathbf{x}_\mathrm{g}) = (\mathbf{x}_\mathrm{s} + \mathbf{x}_\mathrm{g})^T \left[\frac{1}{x_{\max}-x_{\min}} \quad \frac{1}{y_{\max}-y_{\min}}\right]^T + d, \tag{12}$$

where

$$d = \frac{||\mathbf{x}_\mathrm{s} - \mathbf{x}_\mathrm{g}||_2}{\sqrt{(x_\mathrm{max} - x_\mathrm{min})^2 + (y_\mathrm{max} - y_\mathrm{min})^2}}. \tag{13}$$

The random combinations are then ordered according to w and 30 samples are drawn equidistantly from this set to get a path test set that is evenly distributed in space as well as in distance. The derived start and end positions are visualized in Fig. 4.

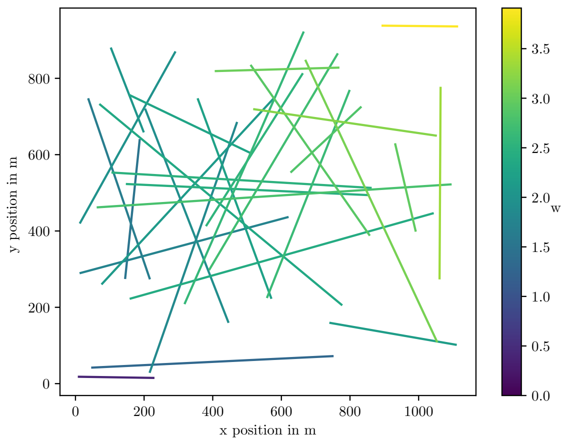

Fig. 4: Start points $\mathbf{x}_\mathrm{s}$ and corresponding end points $\mathbf{x}_\mathrm{g}$ of 30 randomly generated scenarios for the evaluation.

D. Metrics

The solvers are compared on the basis of

- the number of non-dominated solutions $n_\mathrm{non,dom}$ generated by each solver,
- the hypervolume HV of a solver, normalized to the best and worst hypervolumes of all solvers for a certain scenario. The Hypervolume indicator [33] measures the area in the objective space enclosed by the non-dominated solutions and a defined reference point,
- the generational distance (GD) metric by [34]

$$GD(S, S_\mathrm{R}) = \frac{\sqrt{\sum_{\mathbf{s}\in S} \min_{\mathbf{s}_\mathrm{R}\in S_\mathrm{R}} ||\mathbf{s} - \mathbf{s}_\mathrm{R}||_2^2}}{|S|}, \tag{14}$$

  that measures how well the elements $\mathbf{s}$ in the set S of the obtained solutions in the objective space converge towards the nearest solutions $\mathbf{s}_\mathrm{R}$ of a reference set S_R of solutions in the objective space. The lower the GD, the better is the convergence between the obtained front and the reference front.
- the inverted generational distance (IGD) indicator introduced by [35]

$$IGD(S, S_\mathrm{R}) = \frac{\sum_{\mathbf{s}_\mathrm{R}\in S_\mathrm{R}} \min_{\mathbf{s}\in S} ||\mathbf{s} - \mathbf{s}_\mathrm{R}||_2}{|S_\mathrm{R}|}, \tag{15}$$

  which, in contrast to GD, sums up the distances of all reference solutions $\mathbf{s}_\mathrm{R}$ to the nearest obtained solution $\mathbf{s}$. The IGD metric is therefore a measurement for the convergence as well as the diversity of the obtained solutions.

E. Results

The evaluation of the objective functions for all generated solutions is the time-sensitive part of the evolutionary and the gradient-based solver. Accordingly, for a fair basis of comparison, the number of objective function evaluations is fixed to $n_\mathrm{fun,eval} = 18800 \pm 3\%$ for the ES, HES, NS and HNS approach. DIJ and ADIJ do not use function evaluations during the optimization phase. Therefore, we set the number of optimizations for DIJ and ADIJ to $n_\mathrm{MO,sweep} = 65$ so that their measured calculation time lies within the same range of $t_\mathrm{calc} = 92\mathrm{s} \pm 11\%$ compared to the other solvers.

The obtained results can be seen in Table II. It should be noted that the scores generated by DIJ are highlighted in gray, as its solutions only serve as a reference set. They are optimal in the sense of a zig-zag path representation, but they do not meet the requirement of a smooth path that was presented in Section II-B. To meet this condition, the DIJ generated path was smoothed in a post-processing step. This strategy is covered by the ADIJ approach.

The results clearly indicate the weakness of the LB approach for the present problem definition. The reason for that is the multimodal character of the grid maps. The L-BFGS-B solver pushes the paths into local optima and then terminates.

In a similar fashion, the ES and NS approaches will eventually get stuck in local minima. Compared with the reference Pareto fronts of DIJ, this behavior results in the small normalized hypervolumes as well as high GD and IGD values of the Pareto fronts of LB, NS, and ES. As an example, we visualize the Pareto fronts for two of the 30 scenarios in Fig. 5 and Fig. 6. Please note that among themselves the ES approach (green front) outperforms the ES approach (orange front). The automatic step size adaption of the ES might be the reason why more distant solutions are identified more quickly compared to NS, leading to a much more diversified Pareto front.

Looking at the reference DIJ approach, it is interesting to see that it produces only 28 non-dominated solutions from 65 optimization runs with differently weighted objectives. This demonstrates the great drawback of multi-objective problems being solved by a single-objective solver with a weighted aggregation of the objectives. Furthermore, by the smoothing step in the ADIJ approach, the hypervolume drops by 26% from $HV_\mathrm{N,DIJ} = 98\%$ to $HV_\mathrm{N,ADIJ} = 72\%$, losing another 11 non-dominated solutions.

The hybrid approaches HES and HNS compensate for those disadvantages, using our hybrid strategy. They achieve hypervolumes of $HV_\mathrm{N,HES} = 87\%$ and $HV_\mathrm{N,HNS} = 80\%$ and, thus, significantly[1] better normalized hypervolumes than the ADIJ approach in the same amount of time. Furthermore, with the hybrid approaches, the convergence (GD) improves by 12% for the HES and by 5% for the HNS approach as well as the IGD measure by 31% and 15%, respectively.

[1]Applied Wilcoxon two-tailed rank-sum test with $n = 30$, $P < 0.05$ and medians of $M_{\mathrm{HV_N},\mathrm{HES}} = 88\%$, $M_{\mathrm{HV_N},\mathrm{ADIJ}} = 75\%$ and $M_{\mathrm{HV_N},\mathrm{HNS}} = 82\%$ yielding $U_\mathrm{HES,ADIJ} = 732$ and $U_\mathrm{HNS,ADIJ} = 609$.

In Fig. 7, the boxplot for the examined hypervolume metric can be seen. When comparing the HES and the HNS approaches, the first achieves a median of 88%, the latter a median of 82%. By applying the Wilcoxon rank-sum test ($n_1 = 30$, $n_2 = 30$, $P < 0.05$, two-tailed, $U = 609$), it can be found out that the distributions of the normalized hypervolume differ significantly for both solvers.

The paths for the minimum noise extreme points of the Pareto fronts in Fig. 5 are visualized in Fig. 8. Those for the minimum risk extreme points of the Pareto fronts in Fig. 6 are shown in Fig. 9. Looking at the positions of the extreme points of the corresponding paths in the Pareto front plots, it can be seen that in both cases the HES and the HNS solver generate better paths than the ADIJ approach.

TABLE II: Mean results for the evaluation on 30 scenarios

	$\mathbf{n_{non,dom}}$	$\mathbf{HV_N}$	GD	IGD
ES	64	49%	353	461
HES	79	**87%**	**191**	**196**
NS	81	3%	489	719
HNS	**170**	80%	206	245
LB	18	29%	547	606
ADIJ	17	72%	217	284
DIJ	28	98%	9	14

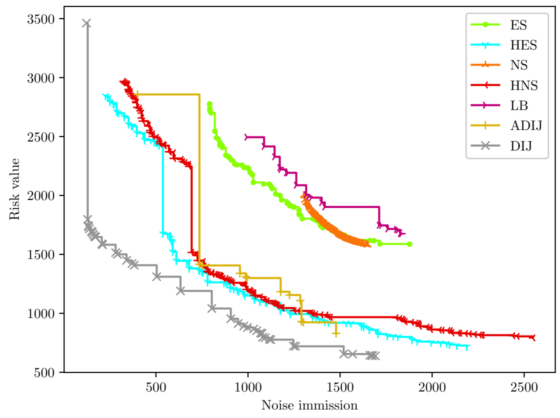

Fig. 5: Comparison of the Pareto fronts obtained by different solvers for one of the test set scenarios (Scenario A).

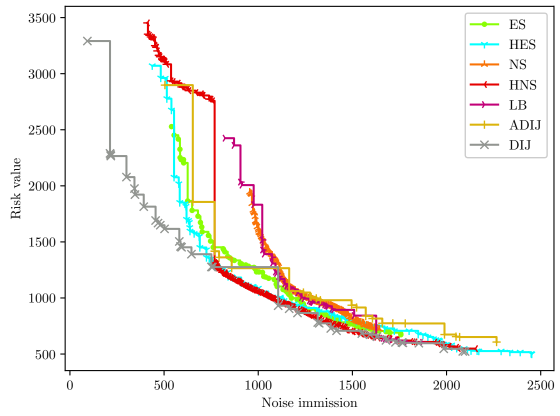

Fig. 6: Comparison of the Pareto fronts obtained by different solvers for one of the test set scenarios (Scenario B).

IV. Conclusion & Outlook

In this work, we formulated a Multi-Objective Path Planning (MOPP) problem and proposed two hybrid evolutionary

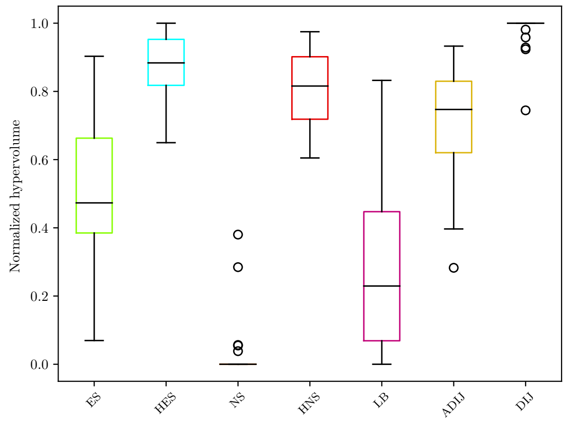

Fig. 7: Boxplots for the different solvers regarding the normalized hypervolume metric.

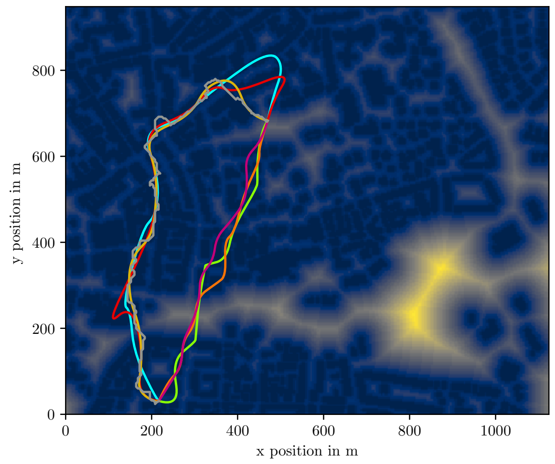

Fig. 8: The best paths calculated by the optimizers for the minimum risk value in scenario A.

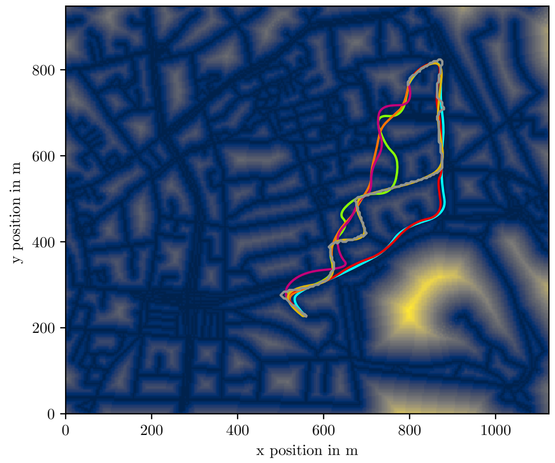

Fig. 9: The best paths calculated by the optimizers for the minimum noise value in scenario B.

algorithms to tackle them. They were benchmarked against state-of-the-art evolutionary, gradient-based and exact algorithms on several instances of a real-world scenario for Unmanned Aerial Vehicles (UAVs). The statistical results of the comparison revealed the advantages of our hybrid approaches.

Exact, graph-based algorithms, like Dijkstra's algorithm, deliver optimal solutions for single-objective problems efficiently. Approximating a Pareto front in multi-objective problems is possible by sampling different weights with a weighted aggregation of objectives. However, this method is inefficient at least in higher dimensions of the objective space, leading to comparably sparse and unequally distributed solutions on the Pareto front. Furthermore, it is noteworthy that the use of specialized algorithms like Dijkstra's algorithm is restricted to objectives whose objective values can be derived from a graph representation. Multi-objective evolutionary algorithms are known to overcome those drawbacks, but require more computational resources. By the advanced initialization step of the proposed hybrid evolution strategy (HES) and the hybrid NSGA-II (HNS) algorithm we combine the strengths of exact and meta-heuristic algorithms. Compared to randomly initialized evolutionary approaches, they gain speed and better convergence towards the true Pareto front. Compared to normal single-objective exact solvers, they gain efficiency in generating non-dominated solutions and thus diversity. Also, the hybrid approaches themselves show a significant difference in performance. The HES approach shows a considerable improvement against HNS in the hypervolume of the generated Pareto fronts. A possible reason might be the step size control of the ES.

In future studies we would like to examine the properties of the different approaches for more than two objectives. For instance, we want to integrate an energy consumption model as a direct objective. The model should include the length of a path on the one hand and its curvature, which should fit to the agent's dynamic motion model, on the other hand. Moreover, as ascending flights of UAVs consume lots of energy, an extension of the representation to 3D seems to be desirable. On top of that it would be interesting to investigate how an adaptation in the number of NURBS control points affects the solution quality, depending on the lengths of the paths.

References

[1] O. Souissi, R. Benatitallah, D. Duvivier, A. Artiba, N. Belanger, and P. Feyzeau, "Path planning: A 2013 survey," in *Proceedings of 2013 International Conference on Industrial Engineering and Systems Management (IESM)*. IEEE, 2013, pp. 1–8.

[2] D. Gelperin, "On the optimality of a*," *Artificial Intelligence*, vol. 8, no. 1, pp. 69–76, 1977.

[3] T. T. Mac, C. Copot, D. T. Tran, and R. De Keyser, "Heuristic approaches in robot path planning: A survey," *Robotics and Autonomous Systems*, vol. 86, pp. 13–28, 2016.

[4] C. A. Tovey, "Nature-inspired heuristics: Overview and critique," *Recent Advances in Optimization and Modeling of Contemporary Problems*, pp. 158–192, 2018.

[5] 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.

[6] R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu, "A limited memory algorithm for bound constrained optimization," *SIAM Journal on scientific computing*, vol. 16, no. 5, pp. 1190–1208, 1995.

[7] E. W. Dijkstra *et al.*, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[8] 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, 2013.

[9] 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, 2013.

[10] Y. Zhang, D.-w. Gong, and J.-h. Zhang, "Robot path planning in uncertain environment using multi-objective particle swarm optimization," *Neurocomputing*, vol. 103, pp. 172–185, 2013.

[11] F. Ahmed and K. Deb, "Multi-objective path planning using spline representation," in *2011 IEEE International Conference on Robotics and Biomimetics*. IEEE, 2011, pp. 1047–1052.

[12] V. Roberge, M. Tarbouchi, and G. Labonté, "Comparison of parallel genetic algorithm and particle swarm optimization for real-time uav path planning," *IEEE Transactions on industrial informatics*, vol. 9, no. 1, pp. 132–141, 2012.

[13] L. Babel, "Three-dimensional route planning for unmanned aerial vehicles in a risk environment," *Journal of Intelligent & Robotic Systems*, vol. 71, no. 2, pp. 255–269, 2013.

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

[15] N. Ganganath, C.-T. Cheng, and K. T. Chi, "Multiobjective path planning on uneven terrains based on namoa," in *2016 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2016, pp. 1846–1849.

[16] S. Jalel, P. Marthon, and A. Hamouda, "A new path generation algorithm based on accurate nurbs curves," *International Journal of Advanced Robotic Systems*, vol. 13, no. 2, p. 75, 2016.

[17] ——, "Nurbs based multi-objective path planning," in *Mexican Conference on Pattern Recognition*. Springer, 2015, pp. 190–199.

[18] Y. Ma, M. Hu, and X. Yan, "Multi-objective path planning for unmanned surface vehicle with currents effects," *ISA transactions*, vol. 75, pp. 137–156, 2018.

[19] J. Rubio-Hervas, A. Gupta, and Y.-S. Ong, "Data-driven risk assessment and multicriteria optimization of uav operations," *Aerospace Science and Technology*, vol. 77, pp. 510–523, 2018.

[20] C. Yin, Z. Xiao, X. Cao, X. Xi, P. Yang, and D. Wu, "Offline and online search: Uav multiobjective path planning under dynamic urban environment," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 546–558, 2017.

[21] A. Goldberg and T. Radzik, "A heuristic improvement of the bellman-ford algorithm," STANFORD UNIV CA DEPT OF COMPUTER SCIENCE, Tech. Rep., 1993.

[22] 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.

[23] L. Mandow, J. P. De la Cruz *et al.*, "A new approach to multiobjective a* search." in *IJCAI*, vol. 8. Citeseer, 2005.

[24] L. Piegl and W. Tiller, *The NURBS book*. Springer Science & Business Media, 1996.

[25] M. G. Cox, "The numerical evaluation of b-splines," *IMA Journal of Applied Mathematics*, vol. 10, no. 2, pp. 134–149, 1972.

[26] C. De Boor, "On calculating with b-splines," *Journal of Approximation theory*, vol. 6, no. 1, pp. 50–62, 1972.

[27] C. De Boor and C. De Boor, *A practical guide to splines*. springer-verlag New York, 1978, vol. 27.

[28] H.-G. Beyer and H.-P. Schwefel, "Evolution strategies–a comprehensive introduction," *Natural computing*, vol. 1, no. 1, pp. 3–52, 2002.

[29] 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, 2013.

[30] R. Cheng, Y. Jin, M. Olhofer, and B. Sendhoff, "A reference vector guided evolutionary algorithm for many-objective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 5, pp. 773–791, 2016.

[31] OpenStreetMap contributors, "Planet dump retrieved from https://planet.osm.org ," https://www.openstreetmap.org , 2017.

[32] D. C. Liu and J. Nocedal, "On the limited memory bfgs method for large scale optimization," *Mathematical programming*, vol. 45, no. 1, pp. 503–528, 1989.

[33] E. Zitzler and L. Thiele, "Multiobjective optimization using evolutionary algorithms—a comparative case study," in *International conference on parallel problem solving from nature*. Springer, 1998, pp. 292–301.

[34] D. A. Van Veldhuizen and G. B. Lamont, "Multiobjective evolutionary algorithm research: A history and analysis," Citeseer, Tech. Rep., 1998.

[35] C. A. C. Coello and N. C. Cortés, "Solving multiobjective optimization problems using an artificial immune system," *Genetic programming and evolvable machines*, vol. 6, no. 2, pp. 163–190, 2005.