drones

MDPI

Article

Improving Motion Safety and Efficiency of Intelligent Autonomous Swarm of Drones

Amin Majd [1,*,†], Mohammad Loni [2,†], Golnaz Sahebi [3] and Masoud Daneshtalab [2]

1 Faculty of Science and Engineering, Åbo Akademi University, Domkyrkotorget 3, 20500 Turku, Finland
2 School of Innovation, Design and Engineering, Mälardalen University, 72218 Västerås, Sweden; mohammad.loni@mdh.se (M.L.); masoud.daneshtalab@mdh.se (M.D.)
3 Department of Future Technologies, University of Turku, FI-20014 Turku, Finland; golnaz.sahebi@utu.fi
* Correspondence: amin.majd@abo.fi; Tel.: +35-844-974-6618
† These authors contributed equally to this work.

Received: 10 June 2020; Accepted: 22 August 2020; Published: 26 August 2020

Abstract: Interest is growing in the use of autonomous swarms of drones in various mission-physical applications such as surveillance, intelligent monitoring, and rescue operations. Swarm systems should fulfill safety and efficiency constraints in order to guarantee dependable operations. To maximize motion safety, we should design the swarm system in such a way that drones do not collide with each other and/or other objects in the operating environment. On other hand, to ensure that the drones have sufficient resources to complete the required task reliably, we should also achieve efficiency while implementing the mission, by minimizing the travelling distance of the drones. In this paper, we propose a novel integrated approach that maximizes motion safety and efficiency while planning and controlling the operation of the swarm of drones. To achieve this goal, we propose a novel parallel evolutionary-based swarm mission planning algorithm. The evolutionary computing allows us to plan and optimize the routes of the drones at the run-time to maximize safety while minimizing travelling distance as the efficiency objective. In order to fulfill the defined constraints efficiently, our solution promotes a holistic approach that considers the whole design process from the definition of formal requirements through the software development. The results of benchmarking demonstrate that our approach improves the route efficiency by up to 10% route efficiency without any crashes in controlling swarms compared to state-of-the-art solutions.

Keywords: safe navigation; evolutionary computing; swarm of drones

1. Introduction

A swarm of drones is a group of autonomously functioning drones providing some services in a coordinated manner. Swarms of drones are increasingly used in a variety of applications such as surveillance systems, goods delivery, rescue operations, etc. [1]. This strongly motivates fast development and deployment of the drone technology. Technology issues are the main factor in drone incidents [2] leading to a raise of serious concerns regarding the safety implications of the technology. Therefore, we should address the problem of ensuring motion safety, i.e., the ability of a system to avoid collisions, while designing autonomous swarm of drones.

In this paper, we propose a novel approach to ensuring motion safety of swarms of drones. Our approach consists of five components including an offline-part, dynamic evolutionary, critical instruction, run-time safety monitoring and decision center. We start by explicitly defining the conditions that should be verified to ensure motion safety of a swarm, which are that swarms do collide with static objects, with each other and/or with the objects that dynamically appear in the fly zone of the swarm. In addition, we consider the route planning as an optimization problem, aiming to maximize

safety while minimizing the length of the path of each drone to achieve higher efficiency. The main novel contribution of this paper compared to a similar solution, DANA [3], is the consideration of dynamic obstacles in the environment with online re-scheduling. The decision center component receives the response of the dynamic evolutionary (default path) and the response of dynamic monitoring (current situation) and checks if we have collisions or not. If the decision center component detects a collision, it sends an interrupt signal to the critical instruction component. The critical instruction component first sends a stop message to all the drones. Then, it notifies the drones to continue their journey one-by-one, so that drones with the highest risk of collision pass first. The dynamic evolutionary component starts to find other efficient routes at the same time and sends them to the decision center component.

In general, path planning is an NP-hard problem on a big search space [3]. Since we aim to optimize multiple objectives at the same time for a couple of agents (drones), the focused problem in this paper is a complex multi-criteria optimization problem. We rely on evolutionary computing paradigm [4,5] to solve such a problem. A combination of a genetic algorithm and imperialist competitive algorithm (MICGA) [5] is the basis of our proposed solution. By mimicking the processes associated with a competition of imperialistic countries to acquire colonies, the algorithm iteratively generates the solutions that progressively maximize (or minimize) the value of the defined fitness function. In our definition of the fitness function, we explicitly introduce safety as an argument, i.e., ensure that our route planning finds the safest shortest route for each drone. We verify that all the routes preserve the defined safety requirements. While safety-aware route planning allows us to handle the predictable hazards, the run-time safety monitoring and control deals with the dynamically emerging hazards. Such hazards are caused by two factors. First, the deviations while executing the planned mission (e.g., caused by drone transient failures). Second, non-deterministic appearance of objects at the fly zone of the swarm. In case of the deviations (the first factor), our solution allows the system to proactively recalculate the routes of the drones to ensure that the swarm continues its mission execution in a safe and efficient manner. In the case of non-deterministic happenings (the second factor), a mechanism for a coordinated collision avoiding maneuvering is activated.

The search space considered in this paper is discretized. A continuous search space provides much more diverse solutions compared to a discretized search space. However, searching a continuous space is slower than a discretized space. We believe that searching a continuous space is reasonable for only one or two drones, while the main idea of this paper is to solve the problem for a swarm of drones. For example, solving the problem with A* is time-consuming for a swarm of drones, therefore, we must limit the search space to provide a real-time search method.

The proposed approach is implemented as a parallel algorithm, which ensures high performance and scalability required for controlling highly dynamic systems. The algorithm is verified to guarantee safety and benchmarked in a number of simulation experiments representing different safety-related challenges (See Section 6). We believe that our approach proposes a novel solution enhancing the motion safety of the autonomous swarms of drones.

Paper Contributions: A summary of the main contributions can be found below:

1. We have defined the main principles of safety-aware route planning for swarms.
2. We have proposed and verified a parallel algorithm that guarantees high performance in solving routing problem for a swarm of drones.
3. We have defined and validated an architecture that combines the route planning with the run-time safety monitoring consisting of proactive collision avoidance and coordinated navigation.
4. We compared our proposed approach with six alternative evolutionary-based navigation solutions including ① Dynamic Autonomous Navigation Algorithm (DANA) [3], ② Dynamic Genetic Algorithm (DGA) [6], ③ Particle Swarm Optimization (PSO) [7], ④ Heuristics and Genetic Algorithms (HGA) [8], ⑤ Rapidly-Exploring Random Trees (RRT) [9], and ⑥ RRT* [10]. The simulation results show our purposed approach achieves 10% and 8.5% improvements on route planning efficiency on average without any crashes compared to the state-of-the-art solutions over Benchmark 1 and

Benchmark 2, respectively. In addition, our proposed approach did not increase the travel distance compared to the optimal safe routing method.

Paper Organization: The paper is organized as follows. In Section 2, we discuss the problem of motion safety and define the conditions that should be verified to ensure collision avoidance. The related works are presented in Section 3. Section 4 presents the principles of the evolutionary algorithms. Section 5 presents the proposed approach and gives small examples illustrating the main steps. We present the simulation results in Section 6. Section 7 concludes the paper.

2. Safety Constraints in Static and Dynamic Navigation

Inspired by [3,11], we formulate the safety requirements of swarm of drones according to the three following criteria:

1. Req1. Preventing the risk of colliding drones with the static objects [3].
2. Req2. Preventing the risk of colliding drones with each other [3].
3. Req3. Preventing the risk of colliding drones with dynamic objects which are not belonging to the swarm such as airplanes [11].

The swarm executes certain missions. A mission is defined as the destinations which should be reached by the drones of the swarm. Each mission of a swarm has two main phases including: planning (offline) and execution (online). In the planning phase, we try to to calculate optimal route for each drone, which is done offline before starting the mission. The goal of the execution phase is to monitor and control the drones during the mission since the drones operate in a dynamic unconstrained environment. By satisfying the mission safety requirements, we can guarantee the behavior of swarm to be compliant to the predefined mission and the possible unexpected situations. Sections 2.1–2.3 formulates these three safety requirements utilized in this paper.

We assume the flying zone is known for each drone during the mission. In addition, the obstacles are defined as the constraints of the planning algorithm. Thus, Req1 is satisfied so that no unsafe routes are planned. $SWARM = \{d_1, \ldots, d_N\}$ is the finite set of drones in the swarm. $AREA = \{l_1, \ldots, l_Z\}$ is the set of locations representing the mission fly zone. M is the number of obstacles located in $AREA$. Obs_i is represented by a subset of $AREA$ which are the occupied locations ($Obs_i \subseteq AREA$). Therefore, $\bigcup_{i=1}^{M} Obs_i$ is all the locations occupied by all the obstacles.$route_i =< l_{in}, \ldots, l_{fin} >$ is a sequence of locations representing a rout for drone d_i ($range\,(route_i) \subseteq AREA$).

2.1. Requirement 1

The safety Req1 can be verified by checking Equation (1) [3].

$$\forall i, j.\ i \in [1, N] \wedge j \in [1, M] \Rightarrow range(route_i) \cap (\cup_{j=1}^{M} Obs_j) = \emptyset \tag{1}$$

2.2. Requirement 2

We need to ensure that each location is occupied by only one drone at the route planning phase. CurrentLoc represents the current position of a drone at a particular instance of time, hence, it is represented as $CurrentLoc = SWARM \times [0..time_{max}] \rightarrow AREA$ where $[0..time_{max}]$ is the time interval covering the entire duration of a mission. $time_{max}$ is the maximum required time for all drones in the swarm to execute their mission. Thus, to avoid drone-to-drone and drone-to-objects collisions, we need to satisfy Equation (2) [3].

$$\forall d_i, d_j, t.d_i, d_j \in SWARM \wedge t \in [0, time_{max}] \Rightarrow CurrentLoc(d_i, t) \neq CurrentLoc(d_j, t). \tag{2}$$

In this problem, we have a certain degree of uncertainty due to drones deviation from the planned routes, e.g., due to a transient fault. In order to model this type of uncertainty, we consider the following scenario. Assume the drones d_1 and d_2 cross at the location l at time t_1 and t_2, respectively,

according the route planning. $\triangle$ is the time gap of reaching the location l by the drones d_1 and d_2 ($t_1 - t_2 > \triangle$). We need to consider $\triangle$ in the route planning in order to ensure safe proximity distance. However, during the mission, d_1 and/or d_2 move slower than expected due to some internal failure, which increases the risk of collision ($t_1 - t_2 < \triangle$).

To dynamically manage emerging safety hazards, we consider the two following actions.

1. To consider safety in the route-planning algorithm. It is achieved by minimizing the number of the cross points and giving a higher preference to cross points with more time gaps between passing drones.
2. we should complement the safety-aware route planning by adding a run-time safety monitoring mechanism. This mechanism monitors the compliance of the predefined routes and schedules with the online status of drones to detect risky deviations. Then, the controlled coordinated cross point passing mechanism will be activated for the detected drones.

All in all, to satisfy the Req2:

1. Safety is considered as one of the route-planning optimization objectives.
2. Safety monitoring dynamically detects emerging hazards.
3. The mechanism of controlled cross point passing is verified to preserve the condition of Equation (2).

2.3. Requirement 3

To address the requirement Req3, we consider two new concerns, including knowledge validity and lack of a prior knowledge regarding the possibility of an object presence [11].

The first concern is explained by the following scenario. In a mission, we have a swarm that flies over the airport take-off zones. Although all the take-offs and landings schedule is known at the route planning time, the actual take off and landing times can change due to the weather or air traffic conditions. Therefore, we have prior information of the location and time of a possible collision locations, but, this knowledge may become invalid during the mission execution.

The second concern discuss a non-deterministic arrival of a flying object, when we do not have a prior knowledge about the appearance of objects or we have a possibilistic prior knowledge. In the case of a possibilistic prior knowledge, we should guarantee the drones will not appear in the predictably occupied zone at the predicted time. Let define $DO = \{obj_1, \ldots, obj_p\}$ as a set of dynamic objects which might appear at the flying zone. The time of appearance of each object and its route are defined by $DoTraj\_f$ function such that $DoTraj\_f : DO \times [1..time_{max}] \rightarrow AREA$. If the positioning of the object is approximate: $DoTraj\_f : DO \times [1..time_{max}] \leftrightarrow AREA$. To guarantee dynamic objects collision avoidance, our route planning algorithm should satisfy the following constraints [11]:

$$\forall d_i, do_j, t.\ d_i, do_j \in SWARM \wedge do_j \in DO \wedge t \in [0, time_{max}] \Rightarrow CurrentLoc(i,t) \neq DOTraj_f(do_j, t) \quad (3)$$

Similarly in the case of approximate positioning:

$$\forall d_i, do_j, t.\ d_i \in SWARM \wedge do_j \in DO \wedge t \in [0, time_{max}] \Rightarrow CurrentLoc(i,t) \not\subset DOTraj_r(do_j, t) \quad (4)$$

As we noted earlier, our priory knowledge is possibilistic. Therefore, the validity of the pre-planned constraints should be confirmed while the swarm is in route. We assume that the planned trajectories remain unchanged, while the time of object appearance might change. To ensure safety, we should guarantee that when any drone approaches a zone that might be occupied by a dynamically appearing object, the autonomous flying mode is changed to the controlled mode if the validity of the pre-planned constraints is not confirmed. In the controlled mode, it should be guaranteed that the drone does not enter the potentially occupied zone until a permission from the air traffic controller is received. We do this by limiting our algorithm to only select the free zone in the search grid. In the case of unavailability

of prior knowledge, we can only rely on the run-time sensor data and trigger controlled mode to calculate a bypass maneuver while satisfying run-time constraints Equations (1)–(4).

In this paper we utilized a 2D motion model (4-neighbourhood or 8-neighbourhood) in order to limit the movement space of each drone. Solving the problem in 2D motion space is more complicated than 3D motion space (6-neighbourhood or 26-neighbourhood) since we have less possible movement options.

3. Related Work

To the best of our knowledge, this paper provides the first algorithm that considers motion safety and efficiency of swarm at the same time, while presenting better results compared to the state-of-the-art solutions. The problem of safe navigation of autonomous robots has been significantly studied in the past [7–10,12–21].

A comprehensive overview of the problems associated with mobile robots is given in [22]. The analysis carried out in [12] shows that the most prominent routing schemes do not guarantee motion safety. A comprehensive literature review on motion planning algorithms for drones can be found in [23,24]. The approaches reviewed in [23] are applicable to a preliminary, offline motion planning phase to plan and produce an efficient path or trajectory for a drone before the start of the mission. Pandey et al. [25] reviews related research trying to solve path planning of autonomous aerial vehicles using meta-heuristic techniques.

Macek et al. [13] proposed a layered architectural solution for robot navigation. They focus on a problem of safe navigation of a vehicle in an urban environment. Similarly to our approach, they distinguish between a global route planning and a collision avoidance control. However, in their work, they focus on the safety issues associated with the navigation of a single vehicle and do not consider the problem of route optimization that is especially acute in the context of swarms of robots. Petti et al. [15] have proposed an approach that relies on a partial motion planning to ensure safety for the navigation of a single vehicle. They state that a calculation of an entire route is such a complex and computationally-intensive problem that the only viable solution is a computation of the next safe states and navigation within them. In our work, to overcome the problem of heavy computational costs and hence insufficiently quick response, we have, on the one hand, discretized the search space, and on the other hand, developed a highly prominent algorithm that guarantees the desired responsiveness. As a result, we could not only calculate entire safe and efficient routes, but also solve this task for a swarm of drones. While defining the overall architecture of the control system, we implicitly structured the behavior of the system using the notion of modes. Indeed, the drones fly in the autonomous mode until a danger of collision is detected. Then, the controlled collision avoidance mode is activated. Barry and Tedrake [16] proposed an obstacle detection algorithm for drones that allows to detect and avoid collisions in real-time. Similarly, Lin [17] presented a real-time path planner for drones that detects and avoids moving obstacles. These approaches are only applicable for individual drones and they do not provide support for a swarm of drones. Bürkle et al. [20] proposed a multi-agent system architecture for team collaboration in a swarm of drones. They also developed a simulation platform for patrolling or surveillance drones which monitor a protected area against potential intrusions. However, they did not address path planning and collision avoidance for the swarm, while we focused on collision prediction, avoidance and efficient navigation of swarms of drones. Augugliaro et al. [21] also presented a planned approach for generating collision-free trajectories for a drone fleet. In contrast to these approaches, our proposed approach combines offline motion planning with a more realistic online route generation approach to produce efficient collision-free routes. Olivieri et al. [26] presented an approach for movement coordination of swarms of drones using smart phones and mobile communication networks. Their work focuses on the internal communication of the swarm and does not provide a solution for collision-free route generation.

Aniculaesei et al. [14] have proposed a formal verification to ensure motion safety. They employ UPPAAL to verify that a moving robot engages brakes and safely stops upon detection of an obstacle.

Since in our work we have focused on finding an algorithm that optimizes safety/efficiency ratio, our solution is more flexible. Our solution also allows the system to dynamically recalculate the route to prevent a collision and avoids unnecessary stopping of drones. Verification of mode logic using the approach proposed in [18,19] would allow us to ensure correctness and safety of the proposed system architecture.

Sujit and Beard [7] proposed a Particle Swarm Optimization (PSO) based anytime algorithm for path planning for a swarm of drones. In their approach, whenever a drone detects a moving obstacle, the anytime algorithm generates a new path for the drone depending on the time allowed to compute a new path before the collision can occur. Although the anytime algorithm avoids collisions with moving obstacles, it works only in certain cases in which the obstacles can be detected from a significant distance. In comparison to the PSO-based algorithm, our proposed approach combines an efficient path generation algorithm with a drone reflex computation algorithm to avoid moving obstacles collisions in various kinds of scenarios. Silva Arantes et al. [8] presented a drone path planning approach for critical situations requiring an emergency landing of the drone. Their approach uses heuristic and genetic algorithms (HGA) to generate and optimize feasible paths under different types of critical situations caused by equipment failures. LaValle [9] presented a sampling based path planning algorithm called Rapidly-Exploring Random Trees (RRT). It was designed to plan efficient paths by exploring high-dimensional spaces and incrementally building a tree. Karaman and Frazzoli [10] proposed an extension of RRT called RRT*. It was designed to asymptotically generate optimal paths by addressing the limitations of sampling-based path planning algorithms.

4. Background: Imperialist Competitive Algorithm (ICA)

Evolutionary computing comprises a set of optimization algorithms, which are inspired by a biological or societal evolution [27]. An example of the former is Imperialist Competitive Algorithm (ICA) [4]. The algorithm simulates a human social evolution. Its parallel implementation [5] shows a remarkable performance in comparison with the other Evolutionary Algorithms (EA) and offers a promising solution supporting computationally intensive tasks of controlling swarm systems. The rest of this section explains the detailed functionality of ICA.

ICA starts by a random generation of a set of countries—the chromosomes (an encoding of the possible solutions)—in the search space of the optimization problem. The fitness function determines the power of each country. The countries with the best values of the fitness function become Imperialists, the other countries become Colonies. The Colonies are divided between the Imperialists and hence the overall search space is divided into empires. An association of a Colony with an Imperialist means that only the chromosomes of the Imperialist and its associated colonies will be used to crossover.

The crossover and mutation are implemented by the assimilation and revolution operators. Mutation is a unary operator applied to a chromosome to produce a (slightly) modified mutant—a child (offspring). Mutation is stochastic, i.e., the child depends of the outcomes of random choices. For instance, a mutation of a chromosome represented by a bit-string can be achieved by a random flip of a bit. Recombination (or crossover) merges the information from two parent genotypes into offspring genotype. Similarly to mutation, the recombination is also stochastic—the choice of parents' chromosome parts and the way of combining is random. Intuitively, a recombination is the of mating two individuals with the different but desirable features to produce an offspring that combines both of those features.

Assimilation moves colonies closer to an imperialist in its socio-political characteristics. It can be implemented by a replacement of a bit of a Colony chromosome by the corresponding bit (or a certain function over such a bit) of the Imperialist.

Revolution is implemented by a random replacement of a certain bit in the Colony chromosome. As a result of assimilation and revolution, a colony might reach a better position and has a chance to take the control over the entire empire, i.e., to replace the current imperialist. This can happen

only if the evaluation of the fitness function of such a colony gives a higher value (if we are solving a maximization problem) than the value of the fitness function of the current imperialist.

The next step of the algorithm is to compute the power of each empire and implement the Imperialistic Competition, which corresponds to the selection of the survival process. The power of an empire is computed as a summation of the value of the fitness function of imperialist and a weighted value of the sum of the fitness functions of the colonies.

The imperialists try to take a possession of colonies of other empires, i.e., the weakest empire loses its weakest colony. Indeed, the weakest empire does not offer a promising solution in the search space and further assimilation of colonies to the current imperialist would not bring any significant improvement. Therefore, it is practical to reallocate the weakest colony to a more promising empire.

In each step of the algorithm, based on their power, all the empires have a chance to take control of one or more of the colonies of the weakest empire. The steps of the algorithm are repeated until a termination condition is reached. As a result, the imperialist of the strongest empire will give us the most optimal solution. The benchmarking simulations demonstrate that the parallel implementation of the proposed algorithm significantly outperforms the similar parallel algorithms. Therefore, it guarantees that at each control cycle the algorithm will generate a desired optimal swarm routing as discuss next.

5. Safety-Aware Routing Planning and Run-Time Safety Monitoring

Our approach to ensuring collision avoidance in the swarm combines parallel ICA-based route calculation with the run-time monitoring. As we discussed in Section 2, the fly area including the positions of the drones can be represented by the set of locations called $AREA$. We assume that the entire fly zone is represented by a grid, i.e., the distances between a pair of neighboring locations are the same, as shown in Figure 1a. The initial and destination positions are known for all drones. The drones move from location to location. Our goal is to find an optimal routing, where routing is defined as a union of each individual drone route, i.e., routing represents a plan of a mission for all drones. We give an ID to each routing and define the set of phenotypes as a set of routing IDs. To explain the principle of defining a chromosome, let us consider an example shown in Figure 1b. Afterwards, we explain the deviation scenarios by considering the example illustrated in Figure 1c.

For the drone d_1 (blue in Figure 1) the shortest path from the initial location to the destination is a sequence <20,19,18,17,16,11,6>, corresponding the shortest paths for the drone d_2 (green in Figure 1) is <21,22,17,12,7,2,3> and for drone d_3 (red in Figure 1) is <11 12,13,14,9,10,15>. We note that the path of each drone can be succinctly represented by a "turning" point—we call it a middle point, which would be 16 for d_1, 12 for d_2, and 9 for d_3. Hence, a chromosome representing such a routing can be represented as a triple $\ll$16,12,9$\gg$. In general, for n drones a chromosome is an $n-tuple$ consisting of the middle points of the corresponding drones. The turning points are generated randomly. To ensure collision avoidance with the static objects, we should explicitly define the locations, which are occupied by the obstacles.

Our route planning starts by generating all shortest paths between each pair of locations in our grid and storing them in a database. The database of the shortest paths is then used to compose the routes of the individual drones as a concatenation of the route from current to the middle point and from the middle point to the final destination. The shortest routes are computed using the algorithm proposed by Dijkstra [28]. For each given source node in the graph, the algorithm finds the shortest path between that node and all other nodes. As an input to our implementation of Dijkstra's algorithm, we define the adjacency matrix of the fly zone $AREA$ with the explicit representation of the obstacles.

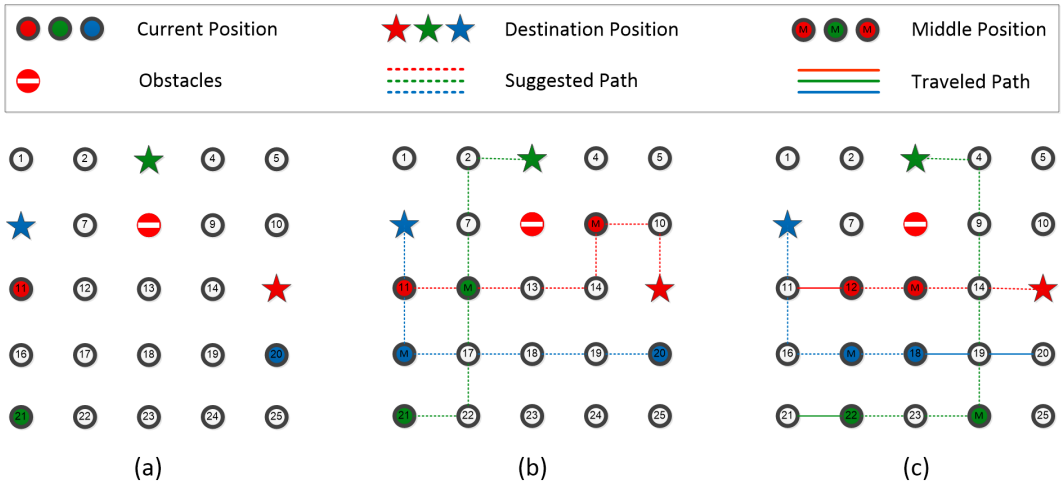


Figure 1. Example of the routing planning. (**a**) Illustration of the search space with the start point and end point for a red drone, a green drone, and a blue drone. (**b**) Illustration of the best suggested path for each drone. (**c**) Illustration of the movement of each drone after the first step.

Now we should define the fitness function to evaluate the fitness of each country (chromosome). As we discussed in Section 2, our goal is to devise an algorithm that optimizes the safety/efficiency ratio. To achieve this, while evaluating fitness of each routing, we should not only evaluate the corresponding path lengths, but also the number of cross points between all drones, as well as the time gap associated with them. The first argument of our fitness function is the distance metric (Equation (5)).

$$Distance\ Metric = \sum_{i=1}^{nd} (Distance_{Current_i \to Middle_i} + Distance_{Middle_i \to Destination_i}) \tag{5}$$

It defines the total length of the drone routes according to the given routing. For our example in Figure 1b, the distance metric of the routing defined by the chromosome $\ll 16, 12, 9 \gg$ is the summation of the lengths of the drone paths: $6 + 6 + 6 = 18$.

The second argument defines the number of cross points associated with the given routing. For our example, the number of cross points is 3 such that in location 17 between routes 1 and 2, in location 12 between routes 2 and 3, and in location 11 between routes 1 and 3, correspondingly.

The third argument of the fitness function is the safety level of the time gap at the cross point. We introduce three safety levels including safety level 0 when there are no cross points, safety level 1 when the time gap at the cross point is above the safety threshold and safety level 2 when the time gap is below the threshold. For instance, in our example shown in Figure 1b, the time gap at cross point 17 is 1 since the drones arrive at that point at times 3 and 1, respectively, and the time gap for the cross point 12 is 2, because the drones arrive there at times 3 and 1, and for the cross point 11 it is 5. As a matter of illustration, we can assume that the time gaps below threshold 2 are classified as safety level 2, while the time gaps at and above threshold 2 as safety level 1. Hence, the cross point 17 obtains safety 2, while the cross points 11 and 12 is the safety level 1.

We define our route optimization task as a minimization problem with the following fitness function (Equation (6)).

$$Fitness\ Function = Distance\ Metric + \alpha \times Number\ of\ Cross\ Point + \beta \times Level \tag{6}$$

α and β are the weight coefficients defined in Equation (7).

$$1 \leq \alpha \leq \frac{nd}{2}, \ 1 \leq \beta \leq \sqrt{np} \times nd, \tag{7}$$

where nd is the number of drones and np is the total number of points. These values allow us to adapt the fitness function evaluation based on the level of complexity of the flying zone and the number of drones. For our example in Figure 1b the value of the fitness function is computed as follows: 18 + 1, 5×3 + 5×3 = 37.5. The evaluation of the fitness function for the initial population is shown in Table 1.

In our large-scale simulation experiments, after evaluating the fitness values of the initial population, we have chosen the imperialists—the countries with the fitness function values smaller than a certain threshold—and the colonies—the other countries. Due to a very small size of the population in our example, we skip this step and explicitly pairwise compare the fitness values. The chromosomes with the lowest fitness values are chosen for crossover and mutation, as shown in Tournament Number and Mating Pool columns in Table 1. The next column defines the probabilities of crossover rc and the results of applying crossover operator are shown in the Offspring after Crossover column. In the similar way, we define the probabilities of mutation. The Offspring after Mutation column shows the results of mutation operator applied to the offspings.

Now we calculate the fitness function for the mutated offsprings. To produce the new generation, from the initial population and the pool of mutated offsprings, we chose the chromosomes with the lowest values of the fitness function. After that, we start the next iteration of the algorithm with the new generation as the current population. After several iterations of the algorithm, we find the routing that achieves our goal—minimizing the distance of travelling and associated danger, i.e., maximizing safety. The pseudocode and flow diagram of the entire approach are shown in Algorithm 1 and Figure 2 respectively.

Let us now illustrate a deviation scenarios. In Figure 1c, we present a snap-shot of drone positions after one unit of time has elapsed. drone 2 and drone 3 have moved according to the planned routes with the planned speed. However, due to some internal problems, drone 1 moved twice as fast as it was supposed to. If drone 1 regains the planned speed and the initial routing is not changed then the drone 1 and drone 2 will collide in location 17. Hence, we should recalculate the routes. This goal is achieved using our proposed algorithm. As shown in Figure 1c, the new routing avoids cross point 17 by rerouting drone 2 to the route <22, 23, 24, 29, 24 , 9, 4, 3> and finding shorter path for drone 3 <12, 13, 14, 15>.

If a collision is predicted between the drones, the priority to move to the next position is given to the drone that is closer to the cross point. Then, after a safe time gap, the next drone moves to the next position and the situation is reassessed. If the collision danger is removed, the routing is recomputed and the autonomous flying mode is resumed.

Table 1. An example of the two iterations of proposed optimization algorithm.

Current Position	Initial Population (Iteration #1)	Total Distance	Cross Point	Critical Level	Fitness	Tournament Number	Mating Pool	r_c	Offspring after Crossover	r_m	Offspring after Mutation	Total Distance	Hot Point	Critical Level	Fitness	Next Generation (Iteration #2)	Best
20	17 24 13	18	4	1	39	1, 3	17 24 13	0.6	16 22 16	0.2	17 24 13	18	6	2	57	17 24 13	16 12 09
21	09 17 18	20	4	2	56	2, 4	16 12 19		17 14 18	0.6	17 14 18	20	3	1	39.5	17 14 18	
11	13 09 23	22	5	1	44.5	1, 4	16 12 19	0.9	16 12 19	0.1	13 09 23	22	4	2	58	13 09 23	
	16 12 90	18	3	1	37.5	2, 3	13 09 23		13 09 23	0.4	16 12 09	18	2	1	44.5	16 12 09	

Current Position	Total Distance	Cross Point	Critical Level	Fitness	Tournament Number	Mating Pool	r_c	Offspring after Crossover	r_m	Offspring after Mutation	Total Distance	Cross Point	Critical Level	Fitness	Next Generation	Best
18	14	1	1	30.5	2, 4	16 12 09	0.5	17 09 09	0.1	17 09 11	16	4	2	52	17 24 13	17 24 13
22	17	2	2	50	1, 3	17 24 13		16 21 14	0.3	16 21 14	14	1	1	30.5	16 21 14	
12	18	4	1	39	1, 4	17 24 13	0.7	18 09 23	0.4	18 09 23	18	3	1	37.5	18 09 23	
	14	2	2	47	3, 2	13 09 23		12 24 13	0.8	12 24 13	14	2	1	32	12 24 09	

Algorithm 1: The pseudocode of the overall algorithm (DE1).

```
1  Function Offline Part():
2      Step 1: Call Dijkstra' Algorithm to Compute the Shortest Path between all nodes
         Step 2: Read Current Position of all Drones.
3      Step 3: Call Generate Countries
4      Step 4: Call Evaluation Operation
5      Step 5: Select the Best Routes
6  Function Online Part():
7      while all Drones has not arrived to their Destinations do
8          if #Crosspoint(BestRoutes) == 0 then
9          while the Best routes and current positions match do
10             only monitoring
11         Call Evaluation Operation
12         else if theDangerouslevel == 2 then
13             Run Critical Navigation Instructions
14             Go to line 5
15         else
16             Call Assimilation and Revolution Operations
17             Call Evaluation Operation
18             Run Competition Operation
19             Go to line 5
```

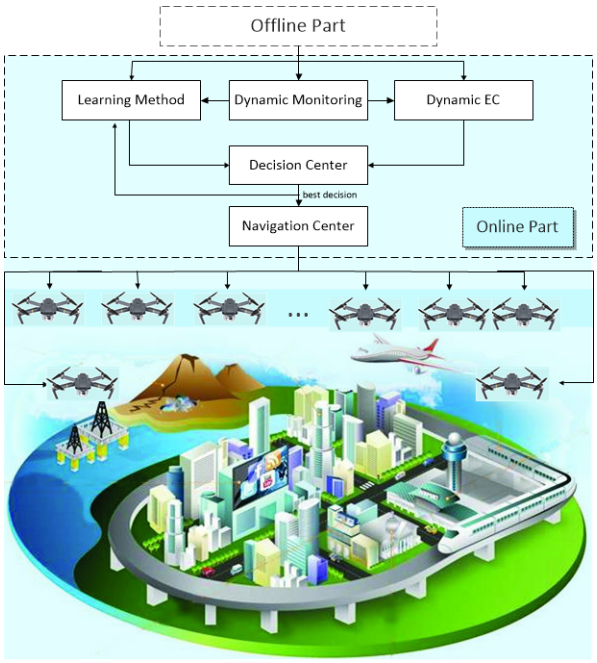


Figure 2. The flow diagram of the proposed algorithm (DE1). DE1 finds an offline scheduling with a safe route for the swarm. At the run-time, our algorithm works properly until the decision center component finds a collision. In the case of collision risk, the navigation center first stops all drones, then activate them one-by-one according to the new scheduling received by the dynamic EC component. Therefore, we can guarantee the desired responsiveness.

The amount of locations does not impact on the run-time of the algorithm. The run-time of the algorithm is influenced by the number of drones since the length of ICA chromosome is the number of drones. Therefore, the search space will be complex by adding more drones.

6. Results

This section presents the simulation results of the proposed approach. The simulation configuration and the characteristics of the evaluation benchmarks are presented in Sections 6.1 and 6.2, respectively. Route length, minimum safe distance, frequency of the route regeneration, number of crashes, and longest route length are recorded as the evaluation metrics. We compared our proposed approach with six state-of-the-art drone controlling methods (Section 6.3). Finally, we analyze the convergency of the proposed optimization approach in Section 6.4.

6.1. Simulation Configuration

We have implemented the proposed approach on a shared memory platform. We used the message passing interface (MPI) to parallelize the proposed algorithm and MPICH2 [29] to run the algorithm. To implement the proposed work, we used four processors in a ring topology. Our algorithm was tested on Intel® Xeon® E5-1620 v3 3.50 GHz processors with 16 GB memory and NVIDIA GeForce GTX 1080 graphics processing units. The configuration of the proposed ICA optimization method is presented in Table 2.

Table 2. The Specification of Imperialist Competitive Algorithm (ICA) configuration.

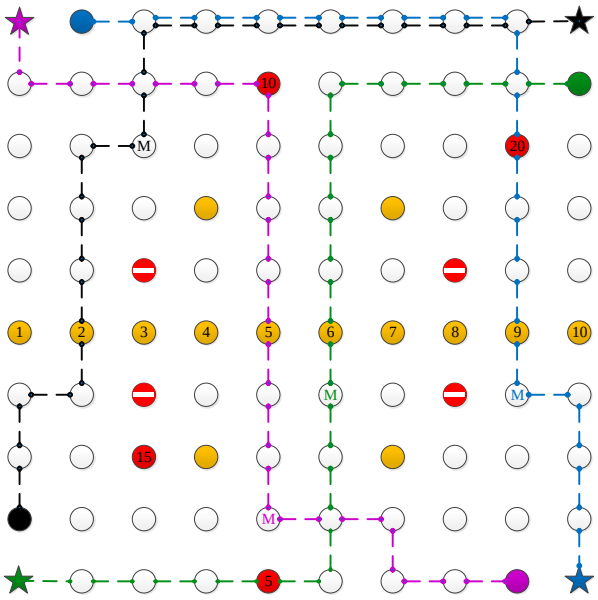
ICA Configuration Parameter	Value
Population size	35
Number of empires	3
Max iterations for each step	10
Assimilation rate	0.8
Revolution rate	0.2

6.2. Evaluation Benchmarks

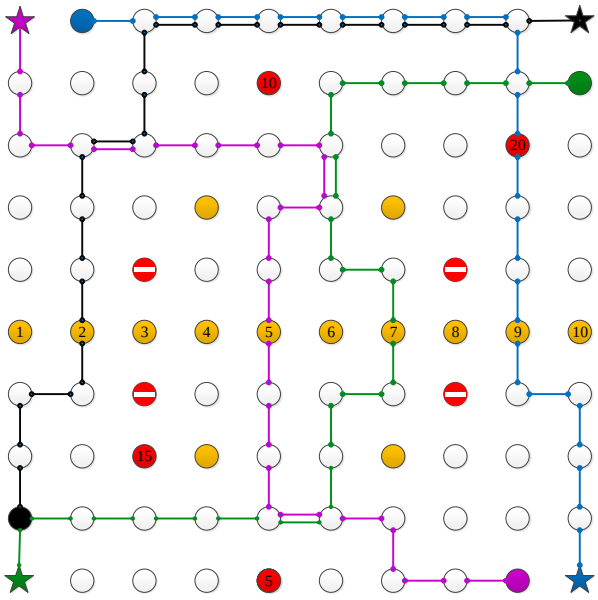
The specification of evaluation benchmarks is presented in Table 3. Benchmark 1 is based on a 10×10 flying zone with 4 drones, 8 static obstacles, 1 dynamic obstacles moving on straight lines from different starting positions, and 4 unforeseen/unpredicted obstacles. Benchmark 2 is based on a 10×10 flying zone with 4 drones, 10 static obstacles, 1 dynamic obstacles moving on straight lines from different starting positions, and 8 unpredicted obstacles. Figures 3a and 4a depict the initial plan for the drones in the first and the second benchmark based on knowledge about the starting points, ending points, static obstacles, and dynamic obstacles. Figures 3b and 4b show the behavior of our dynamic model based on the unpredicted obstacles.

Table 3. Specification of evaluation benchmarks.

Parameter	Benchmark 1	Benchmark 2
Number of Static and Dynamic Obstacles	9	11
Number of Drones	4	4
Number of Unpredicted Obstacles	4	8
Shortest Distance (Optimal) without Safety	43	44
Shortest Distance (Optimal) with Safety	47	48



(a)



(b)

Starting Point
Ending Point
Static Obstacle
Suddenly Obstacle
Dynamic Obstacle
Restricted Area

Figure 3. Illustration of the benchmark 1 flying zone. (**a**) The best suggested path regarding the known static and dynamic obstacles. (**b**) The behavior of algorithm at run-time regarding the suddenly obstacles.

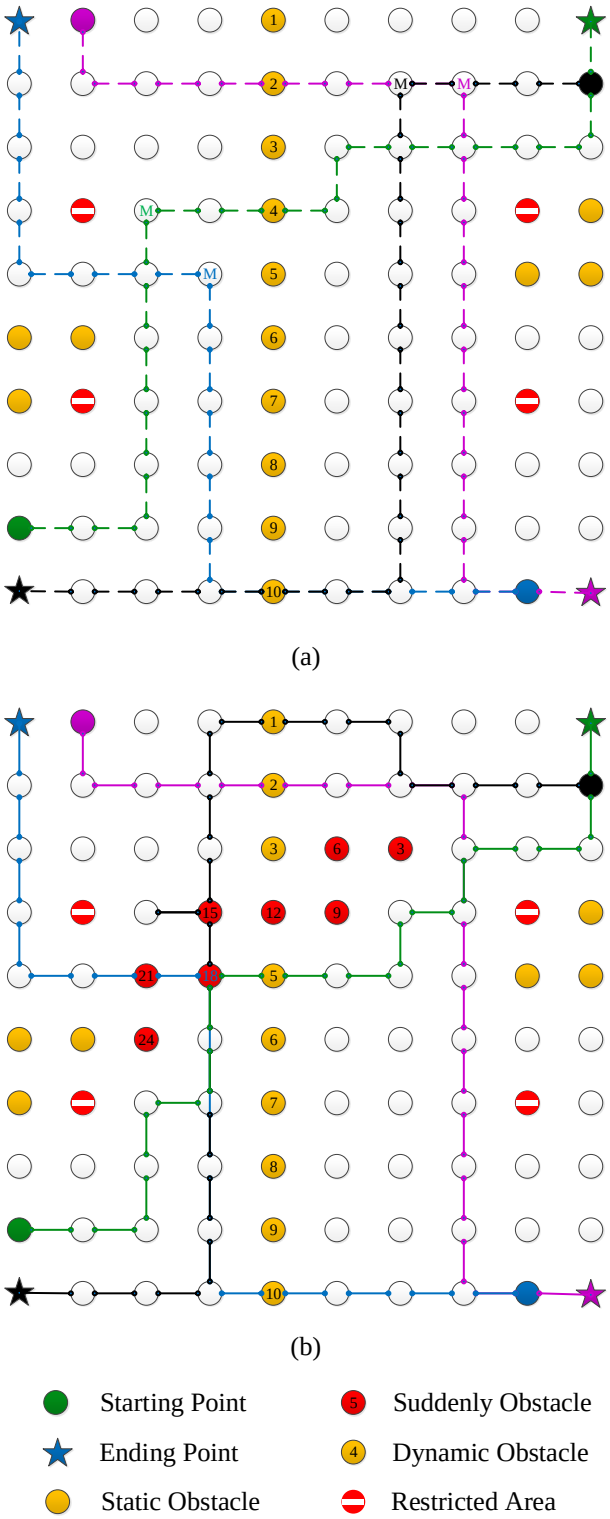


Figure 4. Illustration of the benchmark 2 flying zone. (**a**) The best suggested path regarding the known static and dynamic obstacles. (**b**) The behavior of algorithm at run-time regarding the suddenly obstacles.

The first benchmarks focuses on resolving the problem of the high number of potential cross points and dynamic obstacles, as shown in Figure 3b. The drones should fly in the opposite directions and hence, there is a high risk of collision between each other. The aim of the second benchmark is to

evaluate the algorithms under the challenging flying zone topology for resolving the problem of the high number of potential cross points and dynamic unpredicted obstacles (shown in Figure 3b).

6.3. Comparing to the Other Solutions

We compared our proposed approach, which is called DE1 for simplicity, with six other alternatives including Dynamic Genetic Algorithm (DGA) [6], Particle Swarm Optimization (PSO) [7], Heuristics and Genetic Algorithms (HGA) [8], Rapidly-Exploring Random Trees (RRT) [9], and RRT* [10], which addresses a similar problem. The baseline approach for comparison is called DANA (Dynamic Autonomous Navigation Algorithm) [3]. Tables 4 and 5 compares the results of our proposed solution (DE1) with DGA, DANA, PSO, HGA, RRT and RRT*. The comparison is based on the following seven metrics. The best achieved results with respect to the evaluation criteria are highlighted in the tables using green color.

1. Route length: the length of a drone route measured as the number of steps in the routing. To be minimized to generate shorter routes.
2. Minimum distance: the minimum distance between a drone and an obstacle. To be maximized to generate safer routes.
3. Frequency of route regeneration: the number of times the drone routes are regenerated. To be minimized for reducing the re-computation overhead.
4. Number of crashes: the number of drone collisions. To be minimized to generate safer routes.
5. Length of the longest route: the total number of steps in the generated longest route. To be minimized to generate shorter routes.

The results shown in Tables 4 and 5 demonstrate that DE1 outperformed DANA, DGA, PSO , HGA, RRT, and RRT*. It minimized the route lengths and the number of crashes more efficiently than the others. DE1 generates drone routes with a minimum distance of three steps in benchmark 1 and two steps in benchmark 2, which ensured the safety of the drones. Although RRT generate a new route less frequently compare to DE1, it suffers from more number of crashes because it does not react to the dynamic obstacles.

In benchmark 1, DE1 produced 11.3%, 12.9%, 12.9%, 11.3%, 7.8%, and 4% shorter routes over DANA, DGA, PSO , HGA, RRT, and RRT*, respectively. Similarly, in benchmark 2, it generated 14.2%, 12.7%, 11.1%, 9.4%, 2%, and 2% shorter routes over DANA, DGA, PSO , HGA, RRT, and RRT*, respectively. In addition, DE1 algorithm has successfully and efficiently managed to solve the collision avoidance problem where no collisions occurred with no increased in travel distance (shown in Table 4). The algorithm has succeeded in finding a safe and efficient routing with no increase in travel distance (shown in Table 5).

Table 4. Benchmark 1 results for different studied optimization methods. The best results are shown in green cells.

Comparison Metrics	DE1 (Ours)	DANA	DGA	PSO	HGA	RRT	RRT*
Route length	47	53	54	54	53	51	49
Minimum distance	3	1	2	0	1	0	1
Frequency of route regeneration	5	7	7	6	5	4	5
Number of crashes	0	0	0	1	0	2	0
Length of the longest route	12	14	13	13	14	12	12
Efficiency compared to the optimal route with safety (route length = 47)	100%	88.6%	87.0%	88.6%	87.0%	92.1%	95.9%

The best results are shown in green cells. RRT* means an extension of RRT.

Table 5. Benchmark 2 results for different studied optimization methods. The best results are shown in green cells.

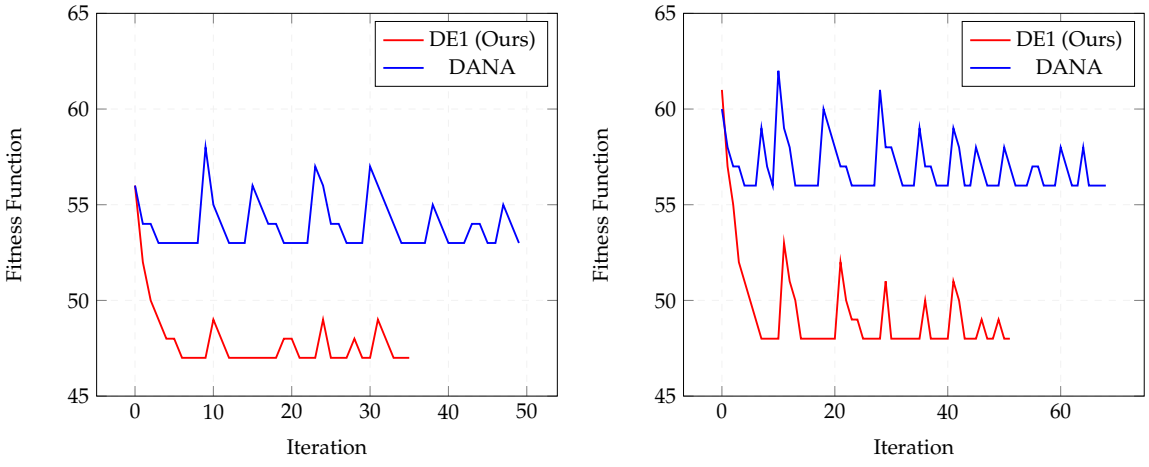
Comparison Metrics	DE1 (Ours)	DANA	DGA	PSO	HGA	RRT	RRT*
Route length	48	56	55	54	53	49	49
Minimum distance	2	0	0	0	0	0	0
Frequency of route regeneration	7	11	9	12	9	4	7
Number of crashes	0	1	1	2	1	4	4
Length of the longest route	12	14	14	13	13	12	12
Efficiency compared to the optimal route with safety (route length = 48)	100%	85.7%	87.2%	88.8%	90.5%	97.9%	97.9%

The best results are shown in green cells. RRT* means an extension of RRT.

6.4. Convergency Analysis

We analyze the convergency of DE1 and compare it with the DANA (Dynamic Autonomous Navigation Algorithm) [3] as the baseline of comparisons. Figure 5a,b depict the convergency variations for the best solution (given to the drone) generated by the DE1 and DANA methods over benchmark 1, and benchmark 2, respectively. Since we have no information of dynamic obstacles, we need to regenerate a new route at the run-time when we face a new obstacle. Each peak in Figure 5 indicates re-computing a new route at run-time due the presence of a dynamic obstacle in the default route generated offline. In a nutshell, we can conclude:

1. Since our proposed approach presents multiple near-optimal solutions at the same time, the number of peaks in the variations of DE1 convegency is less than DANA for both benchmark 1 (40% higher re-computation efficiency) and benchmark 2 (57% higher re-computation efficiency).
2. DE1 needs less computation time to generate a new route compared to DANA, while it presents a collision-free path.
3. The deviation of peaks in DE1 is smaller than DANA since ICA finds three alternative solutions at the same time helping the DE1 to re-converge rapidly.

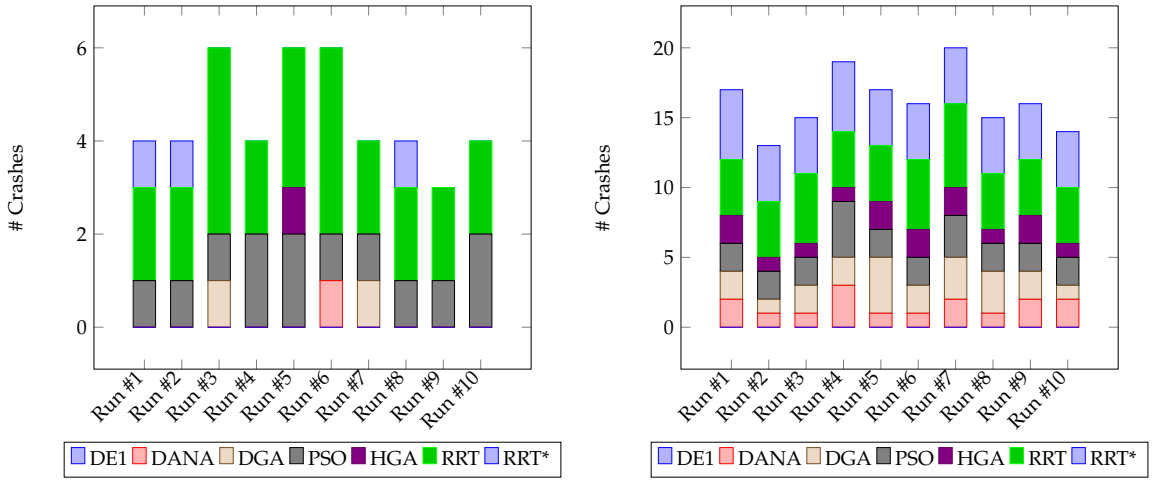


(**a**) **benchmark 1** (**b**) **benchmark 2**

Figure 5. Illustrating the Convergency of the DANA and DE1 over the (**a**) benchmark 1, and (**b**) benchmark 2. DE1 does recalculation at run-time even when we know we have a safe route. The main reason is to improve the efficiency of navigation. Each peak indicates a recalculation.

6.5. Analysing Reproducibility of the Results

To demonstrate that our results are reproducible, we ran all the studied method 10 times. According to the reproducibility results shown in Figure 6, DE1 has no collisions in all the runs, while DANA, DGA, PSO, HGA, RRT, RRT* present 0.1, 0.2, 1.3, 0.1, 2.5, 0.3 and 1.6, 2.2, 2.3, 1.5, 4.4, 4.2 collisions on average over benchmark 1 and benchmark 2, respectively.



(a) benchmark 1 **(b) benchmark 2**

Figure 6. Analysing the Sustainability of different scheduling methods over the (**a**) benchmark 1, and (**b**) benchmark 2.

6.6. Computational Time Analysis

Table 6 compares the computational time of DE1 with other studied methods. RRT and RRT* provides 3.0%, 1.35% (for benchmark 1) and 3.44%, 1.15% (for benchmark 2) faster computation time compared to DE1, respectively. The main reason is that RRT and RRT* are not swarm-based optimization methods and runs the optimization method for each drone separately. Although, RRT and RRT* are faster than DE1, they have 4.4 and 4.2 collisions on average over benchmark 2, respectively, while DE1 presents a safe route.

Table 6. Computational time analysis of the studied scheduling methods.

		Time (Sec.)						
	Time Metric	**DE1 (Ours)**	**DANA**	**DGA**	**PSO**	**HGA**	**RRT**	**RRT***
Benchmark 1	Total time (to the end of route length)	37.62	42.13	97.43	68.44	56.27	12.45	27.85
	Average times of route regeneration	1.41	1.62	2.76	2.54	2.11	1.09	1.12
Benchmark 2	Total time (to the end of route length)	44.16	45.26	99.59	69.18	57.28	12.89	34.28
	Average times of route regeneration	1.63	1.79	2.81	2.67	2.24	1.11	1.28

RRT* means an extension of RRT.

7. Conclusions

In this paper, we have proposed a novel approach to ensuring motion safety of swarms of drones. Our approach relies on the use of evolutionary computing that allows us to formulate safe routing as an optimization problem. A distinctive feature of the approach is its ability recalculate the routing of the entire swarm to maximize safety and efficiency at run-time. To the best of our knowledge, this issue has not been addressed before. We believe that our work has offered a promising solution to the problem of ensuring motion safety of swarms of drones. The benchmarking results have demonstrated that our algorithm is able to manage challenging routing conditions, and guarantees safety while introducing only a small overhead to achieve it.

Author Contributions: A.M. is the driver of the paper that performed implementations, and methodological investigation. M.L. helped on investigation and reviewing and editing the final draft of paper. G.S. helped on validating the results and software developments. M.D. is the thesis co-supervisor and is responsible for funding acquisition. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Swedish Knowledge Foundation (KKS) within the DeepMaker and DPAC projects and Academy of Finland within the CoRa project.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Otto, A.; Agatz, N.; Campbell, J.; Golden, B.; Pesch, E. Optimization approaches for civil applications of unmanned aerial vehicles (UAVs) or aerial drones: A survey. *Networks* **2018**, *72*, 411–458. [CrossRef]
2. Wild, G.; Murray, J.; Baxter, G. Exploring civil drone accidents and incidents to help prevent potential air disasters. *Aerospace* **2016**, *3*, 22. [CrossRef]
3. Majd, A.; Troubitsyna, E.; Daneshtalab, M. Safety-aware control of swarms of drones. In Proceedings of the International Conference on Computer Safety, Reliability, and Security, Trento, Italy, 12–15 September 2017; Springer: Cham, Switzeland, 2017; pp. 249–260.
4. Atashpaz-Gargari, E.; Lucas, C. Imperialist competitive algorithm: An algorithm for optimization inspired by imperialistic competition. In Proceedings of the 2007 IEEE Congress on Evolutionary Computation, Singapore, 25–28 September 2007; pp. 4661–4667.
5. Majd, A.; Lotfi, S.; Sahebi, G.; Daneshtalab, M.; Plosila, J. PICA: Multi-population implementation of parallel imperialist competitive algorithms. In Proceedings of the 2016 24th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP), Heraklion, Greece, 17–19 February 2016; pp. 248–255.
6. Sere, K.; Troubitsyna, E. Safety analysis in formal specication. In Proceedings of the International Symposium on Formal Methods, Toulouse, France, 20–24 September 1999; pp. 1564–1583.
7. Sujit, P.; Beard, R. Multiple UAV path planning using anytime algorithms. In Proceedings of the 2009 American Control Conference, St. Louis, MO, USA, 10–12 June 2009; pp. 2978–2983.
8. Silva Arantes, J.D.; Silva Arantes, M.D.; Motta Toledo, C.F.; Júnior, O.T.; Williams, B.C. Heuristic and genetic algorithm approaches for UAV path planning under critical situation. *Int. J. Artif. Intell. Tools* **2017**, *26*, 1760008. [CrossRef]
9. LaValle, S.M. *Rapidly-Exploring Random Trees: A New Tool for Path Planning*; Iowa State University: Ames, IA, USA, 1998.
10. Karaman, S.; Frazzoli, E. Sampling-based algorithms for optimal motion planning. *Int. J. Robot. Res.* **2011**, *30*, 846–894. [CrossRef]
11. Vistbakka, I.; Majd, A.; Troubitsyna, E. Multi-layered approach to safe navigation of swarms of drones. In Proceedings of the International Conference on Computer Safety, Reliability, and Security, Västerås, Sweden, 18–21 September 2018; pp. 112–125.
12. Fraichard, T. A short paper about motion safety. In Proceedings of the 2007 IEEE International Conference on Robotics and Automation, Roma, Italy, 10–14 April 2007; pp. 1140–1145.
13. Macek, K.; Vasquez, D.; Fraichard, T.; Siegwart, R. Safe vehicle navigation in dynamic urban scenarios. In Proceedings of the 2008 11th International IEEE Conference on Intelligent Transportation Systems, Beijing, China, 12–15 October 2008; pp. 482–489.
14. Aniculaesei, A.; Arnsberger, D.; Howar, F.; Rausch, A. Towards the verification of safety-critical autonomous systems in dynamic environments. *arXiv* **2016**, arXiv:1612.04977.
15. Petti, S.; Fraichard, T. Partial motion planning framework for reactive planning within dynamic environments. In Proceedings of the IFAC/AAAI International Conference on Informatics in Control, Automation and Robotics, Barcelona, Spain, 14–17 September 2005.
16. Barry, A.J.; Tedrake, R. Pushbroom stereo for high-speed navigation in cluttered environments. In Proceedings of the 2015 IEEE International Conference on Robotics and Automation (ICRA), Seattle, WA, USA, 26–30 May 2015; pp. 3046–3052.
17. Lin, Y. *Moving Obstacle Avoidance for Unmanned Aerial Vehicles*; Arizona State University: Tempe, AZ, USA, 2015.
18. Iliasov, A.; Troubitsyna, E.; Laibinis, L.; Romanovsky, A.; Varpaaniemi, K.; Ilic, D.; Latvala, T. Developing mode-rich satellite software by refinement in Event-B. *Sci. Comput. Program.* **2013**, *78*, 884–905. [CrossRef]
19. Iliasov, A.; Troubitsyna, E.; Laibinis, L.; Romanovsky, A.; Varpaaniemi, K.; Ilic, D.; Latvala, T. Developing mode-rich satellite software by refinement in Event B. In Proceedings of the International Workshop on Formal Methods for Industrial Critical Systems, Antwerp, Belgium, 20–21 September 2010; pp. 50–66.
20. Bürkle, A.; Segor, F.; Kollmann, M. Towards autonomous micro uav swarms. *J. Intell. Robot. Syst.* **2011**, *61*, 339–353. [CrossRef]

21. Augugliaro, F.; Schoellig, A.P.; D'Andrea, R. Generation of collision-free trajectories for a quadrocopter fleet: A sequential convex programming approach. In Proceedings of the 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, Vilamoura, Portugal, 7–12 Octorber 2012; pp. 1917–1922.
22. Siegwart, R.; Nourbakhsh, I.R.; Scaramuzza, D. *Introduction to Autonomous Mobile Robots*; MIT Press: Cambridge, MA, USA, 2011.
23. Goerzen, C.; Kong, Z.; Mettler, B. A survey of motion planning algorithms from the perspective of autonomous UAV guidance. *J. Intell. Robot. Syst.* **2010**, *57*, 65. [CrossRef]
24. Kendoul, F. Survey of advances in guidance, navigation, and control of unmanned rotorcraft systems. *J. Field Robot.* **2012**, *29*, 315–378. [CrossRef]
25. Pandey, P.; Shukla, A.; Tiwari, R. Aerial path planning using meta-heuristics: A survey. In Proceedings of the 2017 Second International Conference on Electrical, Computer And Communication Technologies (ICECCT), Coimbatore, India, 22–24 February 2017; pp. 1–7.
26. de Souza, B.J.O.; Endler, M. Coordinating movement within swarms of UAVs through mobile networks. In Proceedings of the 2015 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops), St. Louis, MO, USA, 23–27 March 2015; pp. 154–159.
27. Eiben, A.E.; Smith, J.E. *Introduction to Evolutionary Computing*; Springer: Berlin/Heidelberg, Germany, 2003; Volume 53.
28. Dijkstra, E.W. A note on two problems in connexion with graphs. *Numer. Math.* **1959**, *1*, 269–271. [CrossRef]
29. Balaji, P.; Bland, W.; Gropp, W.; Latham, R.; Lu, H.; Pena, A.J.; Raffenetti, K.; Thakur, R.; Zhang, J. *MPICH User's Guide*; MPI: Winnipeg, MB, Canada, 2014.