IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received November 25, 2021, accepted December 7, 2021, date of publication December 10, 2021,
date of current version December 20, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3134459

Heuristic Drone Pathfinding Over Optimized Charging Station Grid

KEMAL IHSAN KILIC AND LEONARDO MOSTARDA, (Member, IEEE)
Computer Science Division, University of Camerino, 62032 Camerino, Italy
Corresponding author: Kemal Ihsan Kilic (kemal.kemal@unicam.it)

ABSTRACT We proposed a novel optimisation framework for drone-based operations which consists of the optimised Charging Station (CS) grid and the pathfinding heuristics for the drone. The proposed pathfinding heuristics are assessed for two different (triangular and square) CS grid configurations that are optimised for the drone range. We presented the case study of a boat rescue operation that is carried out in the sea. The minimisation of the ''flight distance'' and ''number of chargings'' are the objectives for the drone party and the minimisation of the ''average waiting distance'' (AWD) is the objective for the boat party. We studied the ''single drone with many entities'' case which is a form of Travelling Salesman Problem (TSP). We presented mathematical analysis and simulation results for the effectiveness of the pathfinding heuristic which we called the ''red-grey path'' heuristic. A novel and fast TSP heuristic was also proposed as part of the pathfinding heuristics and its performance was assessed.

INDEX TERMS UAV path optimization, heuristic pathfinding for UAVs, optimized charging station grids for electric vehicles, multi-objective multi-party optimization, TSP heuristics.

I. INTRODUCTION

With today's technology, it is possible to package many electronic devices into drones. The utilization of sensors and communication devices on the drones make these flying devices very versatile, vital, and economic option for many important operations. Among these operations monitoring, search and rescue, industrial inspection, communication, and delivery applications can be listed. The study in [1] presents an extensive survey on the application areas of drones.

Since the drones can fly directly to the desired location without being subjected to any obstacle, they can reach quickly and economically to the desired mission region. A small onboard communication module enables drones to transmit data and receive commands related to the operation. If necessary, they can carry important medical supplies or other necessary items. The main problem for such an operation involving drones is the limited amount of onboard energy available to the drone. This problem can be solved by deploying CSs over the ''mission region'' in optimised grid configurations. Such a CS grid gives an optimised (min number of CSs and no blind-spot) coverage of the mission region. However, for the drone, the path to the targets via CSs should also be optimised for energy-saving. The drone can charge or swap its battery on the individual CSs. In our work, we addressed these problems and proposed heuristics to carry out drone-based missions in an optimised way. Just to give a real-life context to our work, we can consider a consumer-grade drone the DJI Inspire 2 (https://www.dji.com/it/inspire-2). This drone can be seen in Figure 1a.

It has a 4280 mAh battery which can provide a drone 27 min flight. The drone can reach a max speed of 94 km/h achieving about 40 km maximum range. Generally, the battery can be charged in 30 min with linear quick charging. However, the recently emerging technologies reduced the charging time to 5 min [2]. The charging stations can charge the drone with a ''contact-based'' charging system (https://skycharge.de/charging-pad-outdoor) that can be seen in Figure 1b. The charging system has stainless steel landing platform and it is designed for outdoor environments (IP65 grade). The pad provides 500W loss-free charging system for the drones. In our work, the terms ''drone'' and ''UAV'' are used interchangeably. In this context, our work proposes a framework that can help drones to cover and operate on the mission region by deploying CSs in a static grid configuration over the mission region. Although we proposed this framework for general-purpose drone operations, the

The associate editor coordinating the review of this manuscript and approving it for publication was Bijoy Chand Chatterjee.

(a) DJI Inspire 2 drone

(b) The charging pad

FIGURE 1. The consumer grade drone DJI Inspire 2 and the charging station.

''boat rescue'' case study is used as a presentation tool in our article. The choice of the boat rescue case study is also partly inspired by the regional project once attempted in the Marche region of Italy. The project is aborted due to a lack of finances. However, the proposed framework can be easily adapted to ground-based operations. One of the differences between sea and ground-based operations can be CS deployment. On the sea, the CS deployment has more freedom. However, for the ground-based operations, there could be locations that deploying CSs can not be possible. In these cases, the CS grid will have ''holes''. But the pathfinding heuristics can just go around these holes. Another difference can be in the network connection technologies that are used in these cases. For the ground-based operations connection is easier as the coverage is better on the ground. For sea-based operations, long-range connection technologies should be used for drone communication. The current work is an extension of the previous study in [3]. In the previous study, only the CS grid deployment was analysed and the sketches of the pathfinding algorithms were presented. Mostly theoretical aspects were presented in the previous work. Also, the synergy between CS grid deployment and the pathfinding heuristic was not concretely established in the previous work. The current work proposed and implemented a heuristic pathfinding algorithm and presented performance evaluation through simulations and benchmarks. In addition to that, the synergy between CS grid deployment and the heuristic pathfinding is explained through simulations. The degree that red-grey paths provide savings according to the CS grid deployment is measured in the simulations.

We aimed to design a drone dispatcher system by considering multiple static boat configurations and a single mission drone. This dispatcher system can be utilized in an event or time-based simulation. This version of the problem is related to the Traveling Salesman Problem (TSP). On the other hand, the multiple drone version of the rescue problem is related to the generic Vehicular Routing Problem (VRP). However, the fact that off-shore flights are required over a regular geometric CS grid makes it a special and novel variant for both TSP and VRP. In this sense, it can be said that the framework we proposed offers novel contributions to mainstream TSP and VRP research. In addition to that, many concepts and methods can be borrowed from TSP and VRP research. Both TSP tour [4] and VRP [5] are known as $\mathcal{NP}$-Hard problems. For this reason, we proposed novel heuristic algorithms for finding the optimum path for the mission drone.

The proposed framework consists of the deployment of CSs, the novel TSP heuristic algorithm we called ''concaveTSP'', and the energy-saving pathfinding heuristic we called the ''redGreySP'' heuristic (hereafter ''redGreySP'', red-Grey Shortest Path). We presented a mathematical analysis for different CS deployment strategies in connection with the proposed ''redGreySP'' heuristic method. The proposed novel TSP heuristic algorithm is analyzed and benchmarked against the other standard approximation algorithms by considering various standard TSPLIB datasets and custom datasets. The savings from the proposed ''redGreySP'' heuristic are tabulated by comparing the results of the same rescue operation with and without the heuristic. The proposed CS deployment schemes ensure no blind-spot mission region coverage. The proposed TSP heuristic finds the ''best order'' for rescuing the boats. Finally, the proposed ''redGreySP'' heuristic by using a modified shortest path algorithm finds the ''best path'' from one boat that frog-leaps on the deployed CSs to the next boat. The ''graph'' structure that represents the system is dynamically augmented with the red, grey edges when there is a rescue call. After the boat is rescued these red and grey edges are deleted. The mission drone leaves the BS and after completing the rescue mission returns to the BS. In this sense, the TSP tour starts from the BS, visits boats via CSs, and returns to the BS.

The multi-objective and multi-party optimisation is integrated into the proposed framework for the boat rescue case study. While for the drone the shortest rescue (min energy) path is important, for the boats the less AWD is desirable. A weighted scenario-based scheme can be used for these objectives. For urgent missions, the shortest tour for the drone can be completely overridden in favour of the least AWD (or least average waiting time) for the boats. The boats can be prioritized according to the type of the necessary rescue operation. The effectiveness of the proposed rescue framework is evaluated in simulation in which randomly distributed rescue calls are generated over the real-life region over the Adriatic Sea close to the shores of Marche, Italy.

We designed simulations and benchmarks to assess the performance of the proposed concaveTSP heuristic against other standard TSP heuristics. Also, the effectiveness of the proposed redGreySP heuristic is assessed by estimating the savings over the base case (without heuristic). The redGreySP heuristic provides about 17% path length savings for the triangular grid and about 10% for the square grid. While the proposed TSP method is on a par with the standard TSP heuristics in small dataset (less than 1000 vertices) simulations, for the bigger (1000+ vertices) datasets it offers faster solutions. Especially for regular grid datasets, it is better than the other heuristics in every aspect. In its essence, our work

proposes a novel framework solution to the generic problem of region coverage with CSs and a special TSP heuristic for visiting the entities in the covered region in an optimum way. This flexible framework can be used in various drone-based operations like search and rescue, delivery, and monitoring.

The rest of the paper is organized as follows: In Section II we presented related work. Section III presents the elements of the proposed framework in subsections with mathematical analysis and experimental results on the case studies. The basic elements of the proposed framework consist of the CS deployment (Section III-A), the proposed concaveTSP heuristic (Section III-B), the redGreySP heuristic (Section III-C). The experimental results are discussed in Section III-D. The conclusions and ideas for future work are listed in Section IV.

II. RELATED WORK

Drones can reach many places with ease and they can provide economic ways for many important operations. They can be fully automated and can be used in operations that are dangerous for human health. Real-life examples of search and rescue operations can be found in [6]–[8]. A list of advantages of using drones in such operations is presented in [9]–[11]. The paper [12] presents a review of the applications of the drones and details on the types of drones.

Regarding autonomy, the operation modes of drones can be classified mainly into two groups: Autonomous and manual. In the case of autonomous mode, drones are either make decisions by themselves or they communicate with other drones to carry out the mission. When the decision making is from the operator, the manual mode is engaged. The mission can be planned and scheduled centrally by BS or it can be planned in a distributed manner involving swarms of autonomous drones. The work [13] presents a study on the use of swarms of drone related to the ''smart city'' concept. In our research we proposed centrally planned and scheduled mission for the autonomous drone. However, once the drone reaches the operation cite the mode can be changed to autonomous mode for various tasks. Like panning the onboard camera or focusing it on the various locations.

The deployment of CSs is an important design element for the operations involving electric vehicles. Especially for drones since they carry a limited amount of energy, the coverage of the mission region depends on the range of the vehicle and on the CS deployment configuration. ''Refuelling'' techniques can provide partial solution to the energy problem of the drone. However, the operation range/effectiveness of the drone depends on the deployed CS grid. In the case of drone that uses electrical energy, the battery can be charged even replaced with various technologies. The work [14] even suggested ''drone-swapping'' to address the limited onboard energy problem. The charging technologies can be ''contact-based'' or ''contactless''. One example of contact-based technology can be found in [15]. In [16], Wireless Power Transfer (WPT) by using electromagnetic waves is presented as one of the contactless ''charging'' technologies. The Laser-Powered UAV wireless communication system is studied in [17]. While the drone can charge its battery with the contact based technologies [15], An example battery swapping technology is presented in [18]. Beside the cost and deployment/maintenance aspects, these technologies should be evaluated for their ''energy refuelling time'' and ''energy capacity'' that they can offer to the electrically energized drones. Other important aspects are the location and conditions that these technologies can be deployed and operated. The priorities of these parameters are important for the type of operation that drones are utilized for. For example in the case of important rescue operations the ''quick refuelling'' may have the highest priority.

The problem of optimally deploying CSs is studied in [19] and [20]. These studies proposed adjustable CS deployment strategies. As it is stated in [19], the UAV routing research can be grouped into three classes: The delivery truck-drone cooperation, transporting drones mostly with public transportation and deployed CS grid assistance. However, the last group, namely CS grid assistance, can be further classified into two groups based on the research that is done: The CS grid can consist of mobile CSs or static CSs. Our study considers optimum static deployment.

Historically the similar routing/pathfinding problems were studied under the title of ''Fuel Constrained, UAV Routing Problem'' (FCURP). In the seminal work [5], the foundational questions related to the problem is studied by considering cars and gas stations with symmetric travel costs. This framework is extended by considering UAVs and asymmetric travel costs in [21]. The study [22] considered the ''cooperation'' between routing and CS grid in the context of CS mobility. It proposed ''mobile refuelling stations'' and optimal routing related to the CS deployment labelling the problem as ''Fuel Constrained UAV Routing Problem using Mobile Refuelling Stations'' (FCURP-MRS). While these studies considered the generic routing problem, they did not propose synergy between CS grid configuration and the routing algorithms. The operation region is assumed to be completely covered by the CS grid.

Delivery operations with truck-drone systems are getting widely used as drones can fly without much traffic problems. TSP with Drone (TSP-D) is a very widely studied topic. The delivery truck is paired with drones that can serve the customer request. While the truck goes on its way the drone or drones assist the delivery, helping the truck in the TSP tour. The review of such delivery systems and the variants of them can be found in [23]–[28] In our study, a single drone from a single static BS (depot) goes and serves the static boat requests. This is a classical TSP case. However, the necessity of charging for the drone and the specific configuration of the CS grid makes the problem a variant of TSP. In our framework, there is no moving truck but the static BS. The path of the drone is determined by the CS grid configuration. On the other hand, the configuration of CSs should be adjusted according to the range of the drone that is utilized. In this sense, while the boats can be regarded

as ''the clients'', the CSs on the way can be regarded as special clients or depots that should be visited. In addition to these differences, we aimed to evaluate the TSP tour by considering multiple objectives. Especially the consideration of AWD of the TSP tour is a novel contribution we proposed in our framework. Further details and methods on the Multi-Objective TSP (MOTSP) can be found in [29]–[31].

The classical TSP literature is very rich since the problem has been around for a long time. The works [32] and [33] are two seminal works from which many of the heuristic methods were germinated. The studies [34] and [35] are two excellent works that give rather detailed explanations and analyses on the heuristic methods that are used in the TSP approximation algorithms. The TSP, in the graph-theoretic domain, is studied by focusing on the edge distances and vertices. However, the geometric TSP instances contain coordinates of the vertices. This extra information is essential for the approximation algorithms that are based on the geometric configuration of the vertices. Computational geometry offers various tools for ''grouping'', ''connecting'', and ''shaping'' of the vertices on the Euclidean Plane. The convex hull, or in general ''characteristic hull'' methods are useful for fitting a ''shape'' to the points scattered on the plane [36], [37]. In this way, if not all, some of the points can be grouped into a closed curve. This is a useful method for creating an initial sub-optimum tour. In studies [38], [39], the use of convex hull in the context of TSP is discussed. The concave hull is another useful tool that can be used in the same manner [40], [41]. The proposed concaveTSP uses a concave hull in a novel way during the sup-optimum tour creation phase. The ''improvement'' phase in which the created concentric hulls are merged follows this ''construction'' phase. The proposed improvement heuristics are utilized in an ''on-the-fly'' manner during the merging phase.

Our study proposes novel static CS grid deployment strategies and synergistically integrated optimised pathfinding heuristics. By unifying these two research fields in a synergistic way, we believe our study filled a research gap in the drone-related literature. The studies that proposed routing coupled with the CS grid strategy did not elaborate on the analyses on the different CS grid configurations. However, in our work not only do we propose optimum coverage configurations with different geometries, but we also proposed an optimum pathfinding heuristic synergistically adapted to the CS grid configuration. Mathematical analyses were also presented related to the CS grid configurations we proposed. Our study proposed novel metrics for assessing the coverage-effectiveness of the CS grid. Although the proposed framework presented with a case study of boat rescue operations, our study can impact many drone-assisted applications.

III. PROPOSED FRAMEWORK

We presented a brief overview in the Introduction I for the proposed framework. Here we like to discuss the design principles and give a bit of the analysis we have carried out on various elements of the framework. The heuristic algorithms

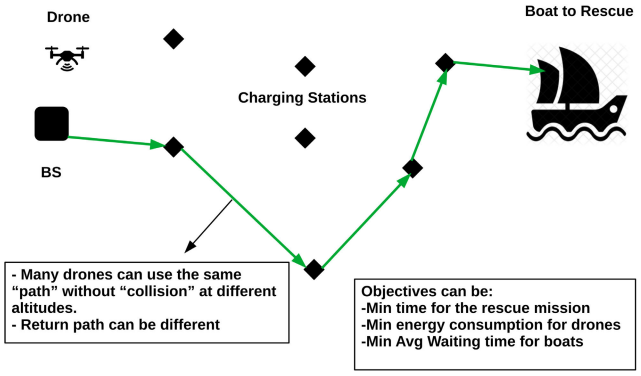


FIGURE 2. Boat rescue operation with drones.

that we designed for optimised pathfinding will be discussed as well. The basic elements of the proposed framework consist of these:

- **Optimised CS grid** by using min number of CSs to cover the mission region without ''blind spots''. Presented in Section III-A.
- The TSP heuristic, **concaveTSP**, for finding the optimum permutation to visit entities (rescue boats). Presented in Section III-B.
- The shortest path heuristic **redGreySP**, for finding the ''best'' path from one entity (boat) to another. Presented in Section III-C.

Simulations and benchmarks are designed to assess the performance of the proposed framework. Design principles and experimental results are presented and discussed in Section (III-D).

Figure 2 shows the representative configuration of the envisioned case study which we called the drone-assisted boat rescue operation.

The physical entities consist of a static CS grid, BS at a static location, and the rescue drone. The mission is centrally controlled by the BS. Upon reception of the rescue request or requests, the optimum path depending on the objectives is estimated and the drone is dispatched for the mission. Currently, the dynamic rescue requests that can be arrived during the flight of the drone are not considered.

For the deployment of CSs, the drone range is the crucial parameter. The CS grid spacing depends on the max drone range and is fundamental for the proposed ''redGreySP'' heuristic. We assumed that the drones can not be charged on the boats. In any case, having a charging facility on the boats does not prevent the utilization of the proposed heuristic. The ''redGreySP'' heuristic searches for a path-length (energy) saving pair of ''red'' and ''grey'' edges on the frog-leaping path of the drone from one CS to another on its way to rescue the ''boat''. The ''grey'' edge is the edge that the drone can fly to/from any CS. The ''red'' edge is the one-way edge for the drone to/from the CS. The ''red-grey path'' is the pair of red and grey edges in which the total distance is less than or equal to the max drone range. Sometimes the drone can find a

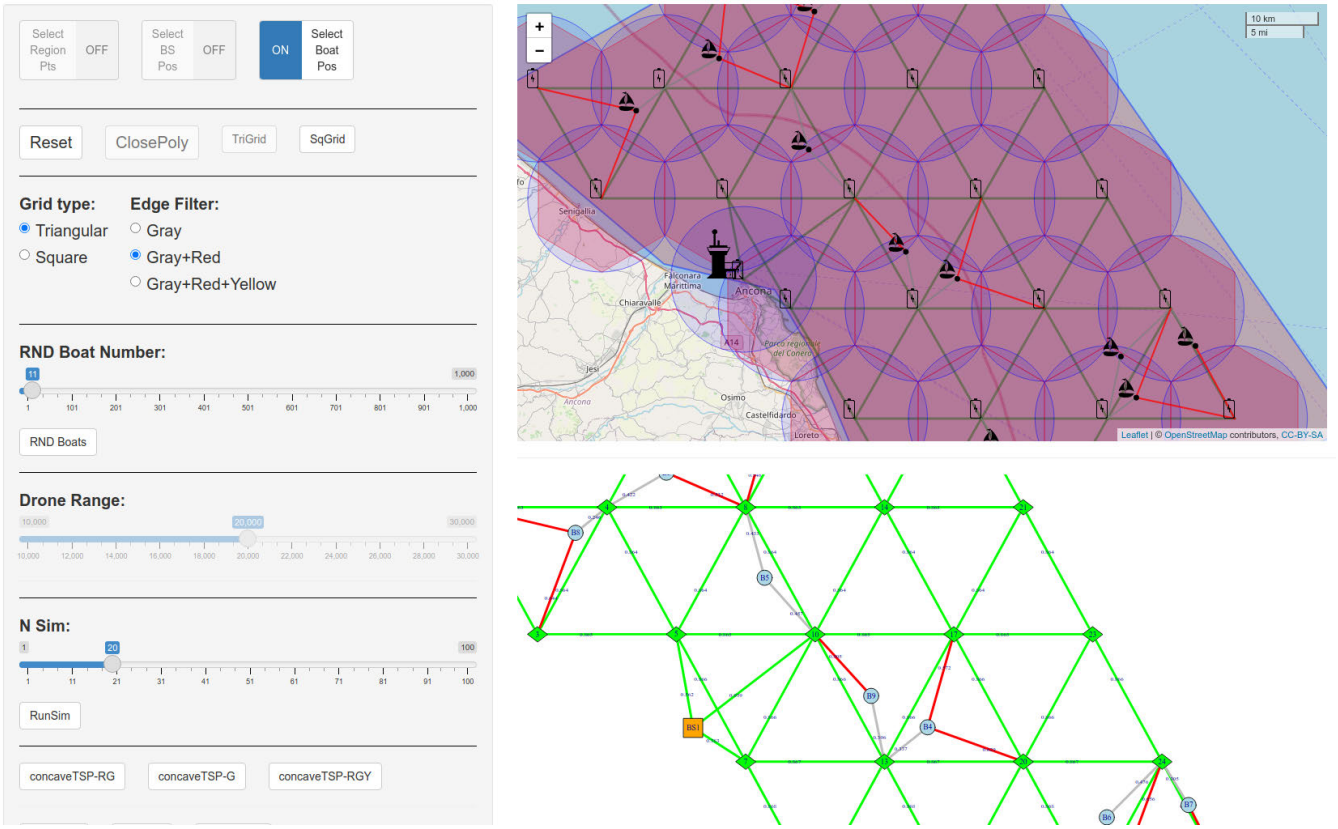


FIGURE 3. Prototype UI developed for the rescue system.

shorter rescue path to the boat when it flies over such an edge pair. The energy budget (battery, fuel tank) constraint of the drone should be checked over such a rescue path to ensure the safe arrival of the drone to the boat without depleting all the onboard energy available. If the CS spacing is not adjusted to the drone range there is a danger of creating "blind spots" in the mission region where boats can not be reached. For this reason, the geometric configuration of the CS grids should be arranged depending on the drone range. The rescue drone should find the shortest TSP tour that starts at the Base Station (BS), visits all the boats, and returns to the BS. On its way, the drone goes frog-leaping from one CS to another charging its battery and utilizing the proposed "redGreySP" heuristic. While the proposed "redGreySP" heuristic considers the shortest path between boats, the custom-designed TSP heuristic algorithm, "concaveTSP", proposes the approximate shortest TSP tour (permutation) for the mission drone according to the current BS and boat configuration. The proposed TSP heuristic algorithm designed for the drone in our framework considers CW (clockwise) and ACW (anti-clockwise) tours for selecting the best rescue path according to tour length and AWD. For our experiments, instead of a weighted scheme, the shortest tour (between CW and ACW) is chosen. In the case of equal tour lengths, the tour with the least AWD is chosen as the best rescue tour. A prototype research software is developed to study the proposed framework in which the user can configure the mission region and select boat positions on a real-world map. The software converts the geographical configuration of the entities to a graph structure and simulates the selected algorithm. The user can see both the geographical configuration and the annotated graph structure used by the heuristic algorithms. A screenshot is given in Figure 3 for this research software. In Algorithm 1 a pseudo-code is given for the proposed rescue framework.

Algorithm 1 The Proposed Rescue Heuristic Framework in Pseudo Code

Input1: ▷ RescuePoly: User drawn polygon containing the rescue region.
Input2: ▷ GridType: User selected CS deployment configuration, tri or sq.
Input3: ▷ DroneRange: User selected drone range.
Input4: ▷ BS and BoatPos: User selected BS and Boat coordinates.
Output: ◁ RescueTour: The optimum rescue tour: $(V_1 \ldots V_K)$

Grid ← NULL
VtxList ← NULL
BS and CS Deployment and Boat position selection
Grid ← Grid + asVertex(userSelect(BS))
VtxList ← VtxList + asVertex(userSelect(BS))
Grid ← Grid + asGraph(deployCS(RescuePoly, userSelect(GridType), DroneRange))
Grid ← Grid + asGraph(userSelect(BoatPos))
VtxList ← VtxList + asVertex(userSelect(BoatPos))
Find the Best TSP tour for the BS + Boats for the drone
concaveTSPTour ← concaveTSP(VtxList)
Find the Best Rescue Path for the drone
RescueTour ← redGreySP(concaveTSPTour, Grid)

Return(RescueTour)

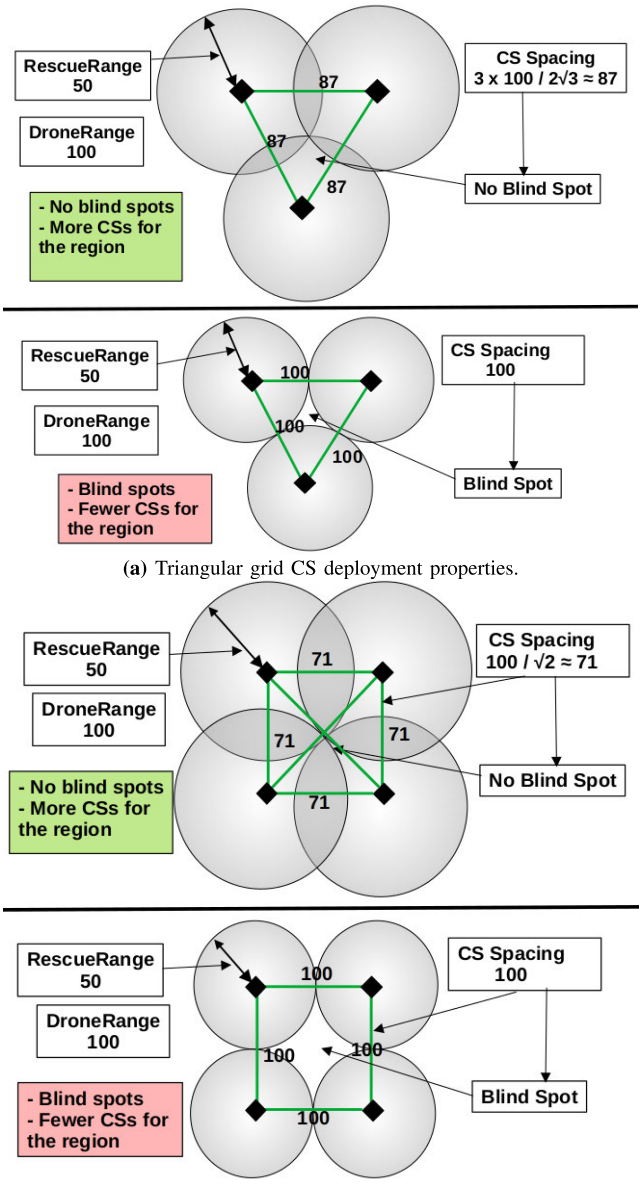
Firstly, the best TSP tour is searched with the proposed ''concaveTSP'' heuristic (2) in the algorithm. For this part, we utilized geometrical distance between vertices (BS + Boats) not the red-grey path distances. In the domain of the red-grey heuristic the edge/path distances are dynamical and direction sensitive. The edges of the red-grey edge/path are removed from the data structures when the associated boat is ''visited''. CW and ACW approaches may not use the same set of red-grey edges or paths. For this reason in finding the approximate TSP tour, we simplified the algorithm with this scheme. This TSP tour is the ''best'' (the approximate shortest TSP tour of the BS + Boats for the drone) rescue order for the boats. Then, with the proposed red-grey shortest path heuristic, called ''redGreySP'' (3), the ''best'' rescue tour is returned. In general, the rescue tour is a multi-objective optimised tour. Namely, the flight distance, AWD, and the number of chargings are the objectives that can be weighted for optimisation. However, for simplicity, in our simulations, we selected the min flight distance by considering the CW and ACW (with respect to BS) versions of the rescue tour. In case of equality, the tour with the min AWD is selected. By default, the algorithm chooses the CW tour in case of equal distance and AWD values. Because of the directional sensitivity of the red-grey edge scheme, CW and ACW tours may differ in total rescue distance. In Algorithm 1, the functions like asVertex(), userSelect(), asGraph(), and deployCS() are used to explain generic procedures. While the asVertex() function selects the vertex of the structures, the asGraph() function selects vertices and the edges of the structures. The userSelect() function represents user interactions via mouse clicks or menu selections. The deployCS() function tries to deploy CSs in an optimum way according to the DroneRange parameter without any ''blind spots'' (inaccessible points for the boats in the rescue region).

A. THE CS DEPLOYMENT

The first step in the construction of the mission infrastructure is CS deployment. For the deployment of CSs so far we studied two different deployment configurations. Namely square grid and triangular grid. To prevent ''blind spots'' in the mission region, CSs should be spaced in a special geometry and according to the range of the drones that are selected for the rescue missions. Throughout our study, we normalized distances according to the drone range unit as it is the fundamental parameter for the proposed rescue framework.

For each case, blind spots should be avoided by estimating the optimum (min number of CSs and no blind spot) inter CS spacing. For any selected geometry for the CS grid, if the inter CS spacing is less than the optimum spacing there will be no blind spots. But more than necessary CSs will be required. If the inter CS spacing is greater than the optimum spacing, then there will be blind spots in the mission region. In Figures 4a and 4b, these situations are depicted.

In Figure 5 actual CS grid deployments are shown on the selected rescue mission region on the sea of Marche, Italy. For the same region, triangular grid coverage requires 24 CSs.

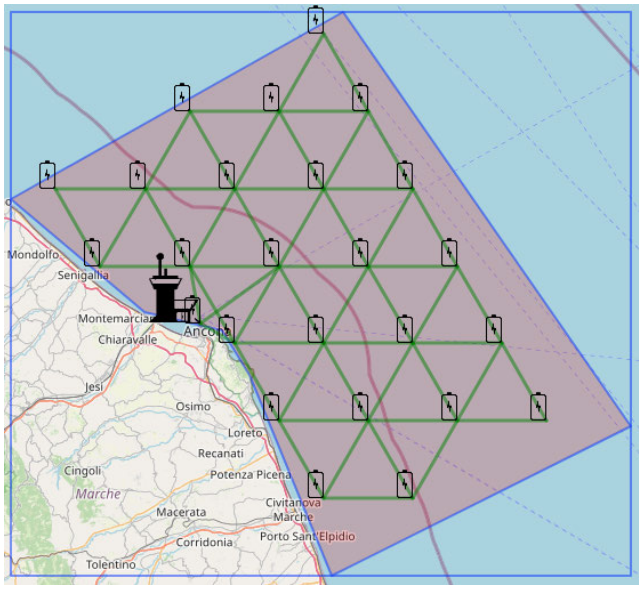


(a) Triangular grid CS deployment properties.

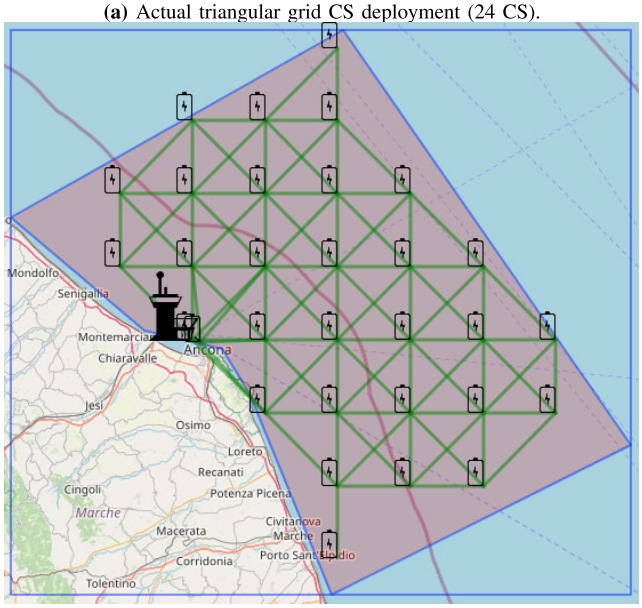
(b) Square grid CS deployment properties.

FIGURE 4. CS grid deployment properties.

For the square grid, the same coverage can be achieved with 30 CSs. The average shortest path length from the BS to all the other CSs for the triangular grid is 2.06 drone range units. For the square grid, the average shortest path length is 2.07 drone range units. The green-coloured edges represent the adjacency. In other words, for the drone, they are possible drone flight paths. The regularity of the CS grid can be exploited in various ways. The proposed ''red-grey path'' heuristic is one of them. In Section III-C we explained in detail how the ''red-grey path'' heuristic can be used to find a better flight path for the mission drone. We also presented mathematical analysis for the savings that can be obtained from the ''red-grey path'' heuristic associated with the selected grid configuration.



(a) Actual triangular grid CS deployment (24 CS).



(b) Actual square grid CS deployment (30 CS).

FIGURE 5. Actual CS grid deployments on the sea of Marche, Italy. The selected region is 5992 km^2. The bounding box has width: 120.93 *km* and height: 109.38 *km*.

B. THE PROPOSED TSP HEURISTIC

The crucial step in the rescue operation is to find the ''best'' order to visit the boats. Considering that the boats have equal priority, the TSP tour of the boats is the shortest tour starting from BS and returning to BS. For this purpose, we proposed a custom TSP heuristic algorithm called ''concaveTSP''. The proposed algorithm can be classified as a hybrid of a geometry-based method with custom ''nearest insertion'' and ''2-Opt-like'' (3-edge) heuristics. The algorithm takes the coordinates of the vertices and produces deterministic output. In the first phase, the proposed algorithm constructs concentric ''rings'' by using the concave hull method proposed in [42]. For the next phase, the constructed rings are merged (if more than one) based on the nearest insertion and 3-edge heuristics we proposed. The concave hull algorithm is based on the geometry of the vertices. The intuition behind using such a sub-tour (concave hull) generation technique was our observations of the actual optimum TSP tours from the TSPLIB (http://elib.zib.de/pub/mp-testdata/tsp/tsplib) datasets. For the optimum tours, we observed, the overall shape was a ''non-self-crossing loop'' [32]. This is logical as self-crossing may introduce some overhead for the tour cost. We imagined modifying the ''outmost'' hull to ''visit'' all the other vertices in a non-self-crossing way. For this task, our first idea was to merge (insert) the remaining vertices into that outmost sub-tour in a systematic way. We think that if we put the remaining vertices onto concentric hulls, this could help us in the neighbour selection process for the merging phase. The levelled concentric hulls sort the vertices according to the distance (and directional order) from the outmost sub-tour and place them into a merging queue. Adjacent concentric concave hulls or rings are very useful for fast neighbour discovery. They kind of play the role of ''geometric priority buffer''. This scheme enables the algorithm to search for ''nearest neighbors'' in a speedy manner.

In Algorithm 2 the pseudo-code is given for the proposed TSP heuristic.

Algorithm 2 The Proposed TSP Heuristic Algorithm (concaveTSP) in Pseudo Code

Input ▷ VtxList: Euclidean vertex coordinates with numbers, (i, X_i, Y_i), $i = 1 \ldots N$
Output ◁ concaveTSPTour: The approximate TSP tour, $(V_1 \ldots V_N)$
Concave hull construction phase
CHList ← NULL
VtxNotVisited ← VtxList
while (|VtxNotVisited| ≥ 2) **do**
    CH ← concave hull(VtxNotVisited)
    CHList ← CHList ∪ CH
    VtxNotVisited ← VtxNotVisited \ CH
end while
RemainedVtx ← VtxNotVisited

Merging phase: Select the nearest concaveTSPTour vertex and use 3-edge heuristic
NSubTour ← |CHList|
concaveTSPTour ← CHList[1]
if (NSubTour ≥ 2) **then**
    for each CH ∈ CHList[2..NSubTour] **do**
        NVtx ← |CH|
        for each Vtx ∈ CH[1..NVtx] **do**
            concaveTSPTour ← Merge(concaveTSPTour, Vtx)
        end for
    end for
end if

Merging RemainedVtx if any
if (|RemainedVtx| ≥ 2) **then**
    concaveTSPTour ← Merge(concaveTSPTour, RemainedVtx)
end if

Return(concaveTSPTour)

The proposed algorithm follows a top-down manner. The inner sub-tours iteratively merged into the outmost sub-tour starting from the outmost inner sub-tour. For each inner sub-tour vertex the nearest ''merged sub-tour'' vertex is selected. We call this method ''nearest vertex merging heuristic''. This selection is faster than the standard ''nearest insertion'' or ''nearest neighbour'' as the nearest vertex is searched

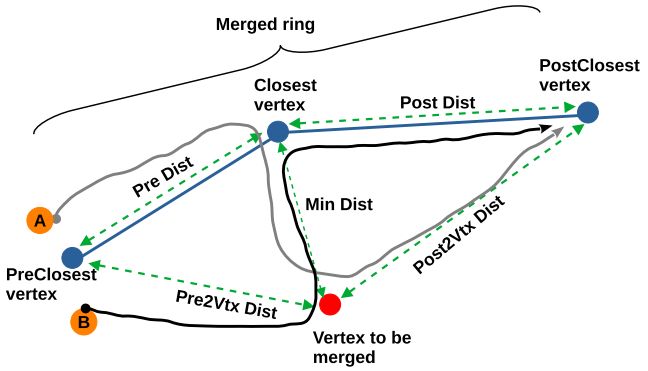


FIGURE 6. The 3-edge heuristic used for merging a new vertex to the ring (sub-tour) merged so far.

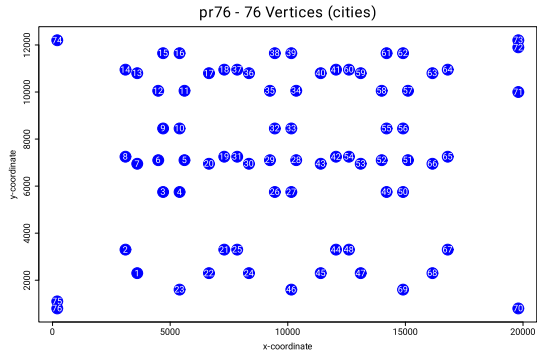


FIGURE 7. "pr76" TSPLIB dataset.

only among the vertices of the merged sub-tour. For this phase, using a distance matrix type buffering speeds up the processing. We also experimented with different vertex selection techniques. Among them, we can list the "nearest edge" and the "nearest edge midpoint". These heuristics are computationally expensive selection techniques. Following the vertex selection, the decision should be made on whether to make the inner sub-tour vertex a successor or a predecessor of the selected nearest merged vertex. For this, we proposed a simple heuristic that considers and compares the cost of three edges shown in Figure 6.

Namely, if the cost of "Pre Dist + Post2Vtx Dist" is less than or equal to the cost of "Post Dist + Pre2Vtx Dist", the path labelled as "A" is chosen. Otherwise, the path labelled as "B" is chosen and the vertex order in the merged ring is updated accordingly. This simple and computationally lightweight heuristic is also based on the geometric positions of the vertices. Functionally, it is a form of the on-the-fly 2-Opt method. We call it a "3-edge heuristic". At the final stage, if there are any, the vertices that do not belong to any sub-tours are also merged in the same way.

According to the paper [42], the time complexity of the concave hull generation for N vertices is dominated by the term $\mathcal{O}(N \log N)$. However, we generated hulls one after another removing the vertices that are already used in the previous hulls. The number of iterations (number of rings) depends on the geometric positions of the vertices. If we use k for the necessary number of iterations (generating k rings) to exhaust all the vertices up to two vertices for ring generation, then the time complexity of the construction phase becomes $\mathcal{O}(kN \log N)$. For the second phase which is the merging phase, again the time complexity depends on the number of the rings generated in the previous phase and on the number of the vertices on these rings. However, assuming a linear search, $\mathcal{O}(N)$, for the nearest vertex, the time complexity of the merging phase is bounded by $\mathcal{O}(cN^2)$, where c is a constant value $0 < c < 1$. If more than one ring is created then each iteration of the second phase passes a fraction of the total N vertices, but never all of them since, for each vertex merging, the nearest vertex selection search considers only the previous ring vertices but not all of the vertices. Basically, every time there are more than one ring, a fraction of the all vertices will be on the merged ring ($\mathcal{O}(rN)$, $0 < r < 1$) and other fractions will be on other rings ($\mathcal{O}((1-r) \times N)$, $0 < (1-r) < 1$). For the merging, every inner ring vertex is paired with a vertex on the ring merged so far ($\mathcal{O}((r-r^2)N)$, $0 < r < 1$). We can think of the worst-case in which two rings are created and each having $\frac{N}{2}$ vertices ($r = \frac{1}{2}$). Then, for each of the inner vertices, the other outmost ring vertices will be checked. This gives us roughly $\mathcal{O}(\frac{N^2}{4})$ processing. As a result, in the worst case, the algorithm is bounded by the merging phase with time complexity of $\mathcal{O}(N^2)$. The merging phase can be further optimised computationally if we use a type of data structure that restricts the search to a specific region or radius. Instead of linear search some kind of priority queue like "minheap" can be used for an effective minimum search with $\mathcal{O}(logN)$ time complexity cost (for insertion). So, the proposed algorithm can offer $\mathcal{O}(N \log N)$ time complexity.

The performance of the algorithm is assessed against similar TSP heuristics and the results are tabulated in Section (III-D). The proposed TSP algorithm is generally fast. We observed that it is better than the other similar TSP heuristics in all the metrics we studied when the "cities" are in a big (1000+ vertices) regular square or triangular grid. For a better understanding of the working of the proposed algorithm, we presented several graph plots that show the various running stages of the algorithm.

Figure 7 shows the standard TSPLIB dataset "pr76" (http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsp/pr76.tsp) with 76 vertices labeled with red colour.

Figure 8 shows the generated 2 "rings" by concave hull with different colors. The vertices are numbered with the standard TSPLIB numbers (blue, on top of the vertices) and also with the ring-specific numbers (black, on the vertices). The outer ring with pink colour is the first ring and the inner ring with green colour is the second ring.

Figure 9 and 10 show the process of merging the first and the last point from the second ring respectively.

In Figure 9 vertex number 20, the first vertex of the second ring is paired with the vertex number 5 (ring 1 vertex 19, coloured with blue) of the merged (the first) ring. The next vertex in the merge buffer is vertex number 9 (ring 2 vertex 2) which is coloured with red. In the last merging round

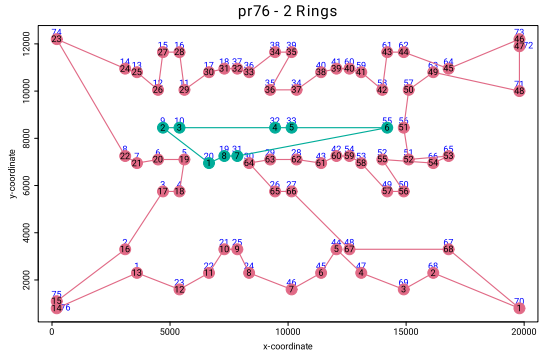


FIGURE 8. Rings generated by concave hull for pr76.

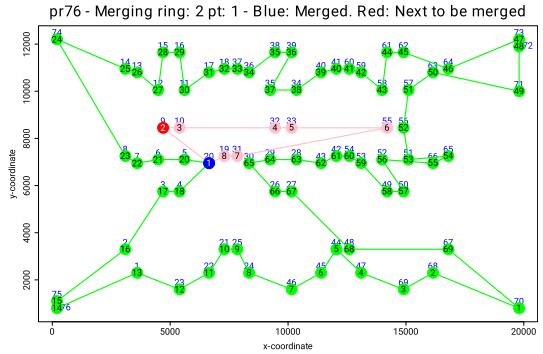


FIGURE 9. Merging the first point from the second ring. The next ring marked with red colour.

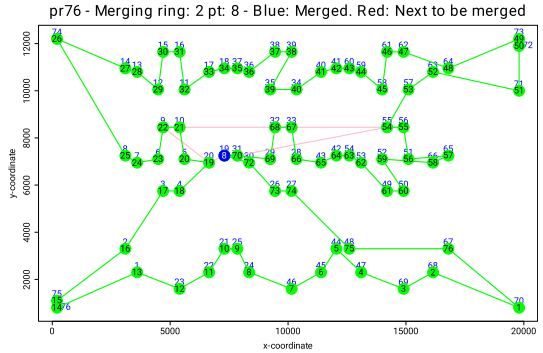


FIGURE 10. Merging the last point from the second ring.

vertex number 19 (the last vertex of the second ring) is merged to vertex number 31 which was previously merged to ring 1 (Figure 10). Figure 11 shows the final ''merged ring'' (the approximate tour from the proposed algorithm) in green colour. The 3-edge heuristic worked well for the vertex chain 6-9-10-5. However, the vertex chain 29-31-19-30 was not a very cost-efficient path. Lightweight post-processing can be carried out to improve such paths. The optimum tour is 1-76-75-2-3-4-5-6-…-24-25-21-22-23-1, which is marked with a dashed line.

C. THE PROPOSED RED-GREY PATH HEURISTIC

Assuming that the drone does not do charging on the boats, after the rescue operation it should have enough energy to

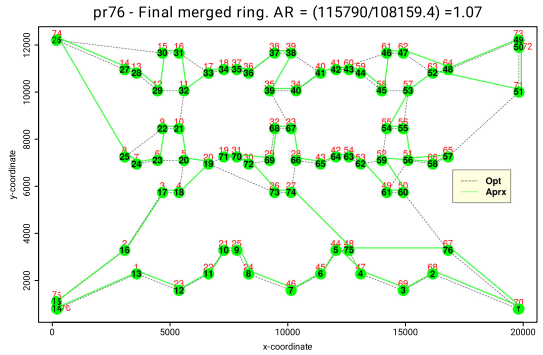


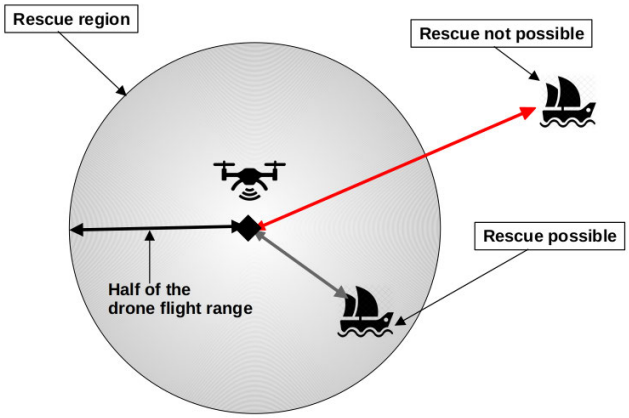
FIGURE 11. The final merged ring and the optimum tour for pr76.

reach any CS. Otherwise, the mission can not continue and the drone can not return to the BS. The drone can safely return to any CS that is in the 50% of its range of any boat. The optimum CS configuration guarantees that any boat anywhere can be reached in this way. Any ''edge'' between any boat and any CS that is less than or equal to half of the drone range is called ''grey edge''. The drone can safely go, rescue the boat, and come back to the CS or go to another CS. Every boat is guaranteed to have at least one grey edge in the CS grid. If there is an edge greater than half of the drone range but less than the drone range is called ''red edge''. The drone can go and rescue the boat but can not return on the red edges. However, in some situations, the combination of ''red-grey'' edges can make the rescue operation possible if the sum of their lengths is less than or equal to the drone range. We call such an edge pair ''good red-grey path''. These conditions related to drone range are explained in Figure 12.

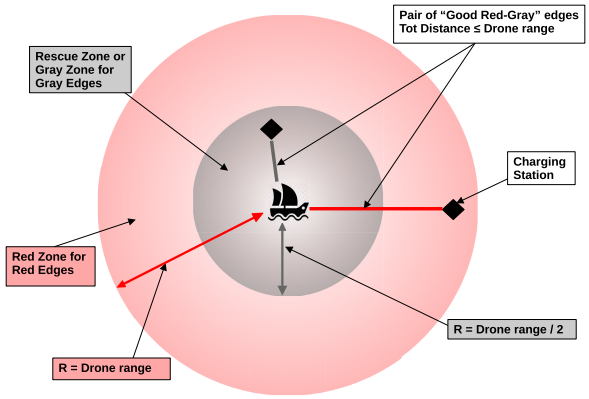
In Figure 13 the situation of ''good red-grey path'' is depicted.

The red-grey path edges are dynamically added to the graph structure as the boats make rescue calls. This red-GreySP heuristic is a general heuristic that can be integrated into any optimum path-finding algorithm like Shortest Path and A*. After the good red and grey edges are added to the graph structure, the path-finding algorithm can just find the optimum path to the boat. However, whether the final destination of the path is a boat or the CS after the grey edge should be considered for the next traversal. If the red edge is used the drone should find and use the grey edge to reach the nearest CS. If it gives a shorter path, the drone can use a red edge after the grey edge. But the red edge should end up at a CS.

Assuming that the boats can ask for rescue at any point on the grid with equal probability, we can present an analysis on the frequency of having such a red-grey edge pair. This analysis can justify the importance of using such a heuristic and can verify the effectiveness of the grid coverage. Figures 14a and 14b helps us to understand the analysis focused on the triangular and square regions among different CS grid configurations. The CSs are at the vertices of the triangles and squares. The circles represent the rescue region of the drone from the CSs. Any point in the circle of the



(a) Conditions for rescuing a boat.



(b) Red-gray edges relative to the boat position.

FIGURE 12. Rescue conditions and red-grey edges.

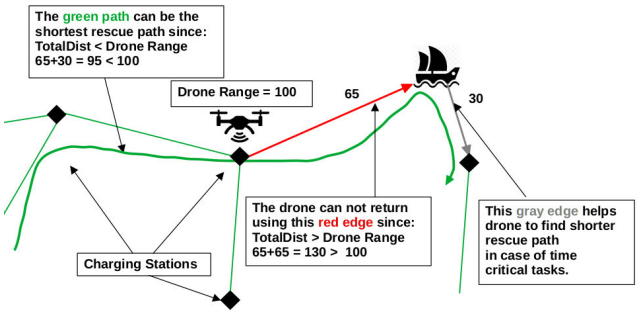
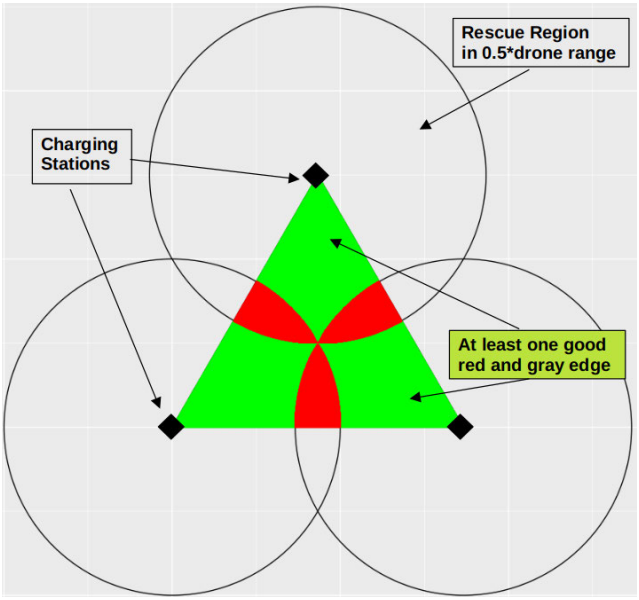
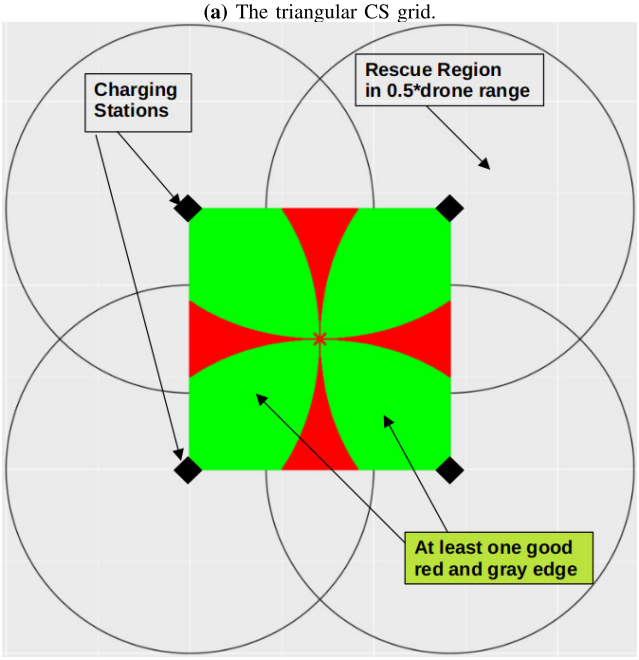


FIGURE 13. The good red-grey path making the rescue possible.



(a) The triangular CS grid.



(b) The square CS grid.

FIGURE 14. The prob. of having a "good red-grey path" (the green region) in different CS grid types.

associated CS has a grey edge leading to it from the station. Outside of the circle points are connected with the ''red'' edge to the associated CS. The green areas represent the locus of the rescue points with ''good red-grey paths'' (the sum of distance is less than or equal to drone range) pairs. The red regions represent the locus of the ''bad red-grey'' edges (when a total distance is greater than the drone range). The probability of having a good red-grey path is simply the ratio between the green area and the total area of the (red + green). A point sampling is carried out in the triangular and square regions. Then for each sampled point, the type of the point and the edge distances to vertices are analyzed.

On Tables 1 and 2 point statistics are presented for triangular and square grids respectively. Small letters ''b'' and ''g'' represent bad and good respectively, whereas capital ''R'' is for Red and ''G'' is for Grey. ''1gR+1G+1bR'' means the point that is connected to vertices with 1 good Red edge, 1 Grey edge, and 1 bad Red edge. The total number of points marked with ''(*)'' gives the total number of points with at least one good red-grey edge pair. The probability of having at least one good red-grey path is the ratio between the total number of points with at least one good red-grey edge pair

TABLE 1. Point statistics for triangular grid. 69483 points are sampled.

Point Type	N	Prob.
3bR	0	0
1G+2bR	0	0
2G+1bR	15171	0.218
3G	5	0.00007
(*)1gR+2G	68	0.00098
(*)1gR+1G+1bR	14428	0.208
(*)2gR+1G	39811	0.573

TABLE 2. Point statistics for square grid. 160801 points are sampled.

Point Type	N	Prob.
4bR	0	0
1G+3bR	0	0
2G+2bR	29452	0.183
3G+1bR	64	0.0004
4G	5	0.00003
(*)1G+3gR	736	0.0046
(*)2G+2gR	0	0
(*)1G+2gR+1bR	66752	0.415
(*)3G+1gR	64	0.0004
(*)2G+1gR+1bR	63728	0.396
(*)1G+1gR+2bR	0	0

and the total number of points. For the triangular CS grid, sampling with 69483 points gives us 0.78 and for the square CS grid, sampling with 160801 points gives us 0.82. This means for each rescue call, about 78% of the time for the triangular grid and about 82% of the time for the square grid, we can have a good red-grey edge pair that can give us a better shortest path for the mission. Because of the rounding and choice for considering boundary conditions with equality or without equality (when comparing the distances), the numbers in Tables 1 and 2 contains some errors.

After finding the probability of having a red-grey edge pair we need to find out the probability of benefiting from this heuristic. For this, the direction of the drone and the direction of the red-grey path should be aligned in a special way in order to benefit from this heuristic. If the drone meets the grey edge first, it will just save the boat and return to BS via that grey edge. But, if the drone meets the red edge first, it will utilize the red-grey path with savings and return to BS. Figure 15 summarizes all possible directions drone can approach the red-grey edge pair and the outcome of these directions for the triangular grid. A similar analysis can be made for the square grid. Now we can estimate the probability of benefiting from the redGreySP heuristic by using data from Tables 1 and 2. Firstly, we should have a "good direction" (gD in Equations 1 and 2) for the boat in which there is a good red-grey path. For the triangular grid Figure 15 suggests that we can have $\frac{1}{3}$ probability to choose a direction for

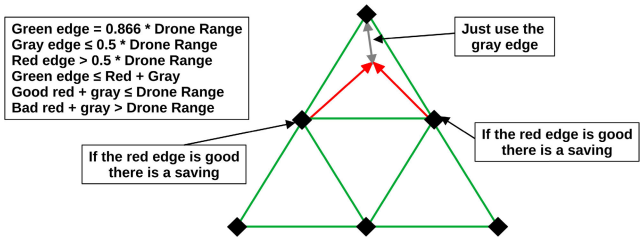


FIGURE 15. The coming direction of the drone towards a "good red-grey path" in the triangular CS grid.

TABLE 3. Edge statistics for triangular grid. 208449 (3 edges * 69483 points) edges are sampled.

Edges	N	Prob.	Min	Max	Avg
gRG	94118	0.68	0.866	1	0.914
bRG	44838	0.32	1	1.253	1
All-RG	138956	1	0.866	1.253	0.968
gR	94118	0.45	0.502	0.866	0.652
bR	29599	0.14	0.502	0.779	0.677
All-R	123717	0.59	0.502	0.866	0.658
gG	54307	0.26	0	0.496	0.271
bG	30425	0.15	0.364	0.502	0.448
All-G	84732	0.41	0	0.502	0.334
All	208449	1.00	0	0.866	0.527

the drone in the case of the triangular grid when we have a single good Red edge (1gR). In Equation 1 the probability of benefiting from the red-grey path is given as 0.45. Similar to the triangular grid case, for the square grid the drone can approach the good red-grey edge pair from one of the 4 directions. In Equation 2 the probability of benefiting from the red-grey path is given as 0.31.

$$\begin{aligned} P(Benefit_{Tri}) &= P(1gR) \times P(gD) + P(2gR) \times P(gD) \quad (1)\\ &= 0.21 \times 0.333 + 0.57 \times 0.666 = 0.45 \end{aligned}$$

$$\begin{aligned} P(Benefit_{Sq}) &= P(1gR) \times P(gD) + P(2gR) \times P(gD)\\ &\quad + P(3gR) \times P(gD) \quad (2)\\ &= 0.3964 \times 0.25 + 0.415 \times 0.5\\ &\quad + 0.0046 \times 0.75 = 0.31 \end{aligned}$$

Finally, we need to know the actual savings from this heuristic. For this we need to consider the edge statistics given on Tables 3 and 4.

The edge length metrics on these tables are normalized according to the drone range. On tables, "gG" (good Grey) refers to the grey edges that can be part of any red-grey edge pair and "bG" (bad Grey) edge type refers to the grey edges that can not be part of any red-grey edge pair. In short, if a pairing (total edge distance less than nor equal to Drone Range) is possible, both red and grey edges become "good".

The actual savings depends on the path length, the number of boats that can be saved in a single tour, and other factors. However, for the actual savings, we can analyze the last part

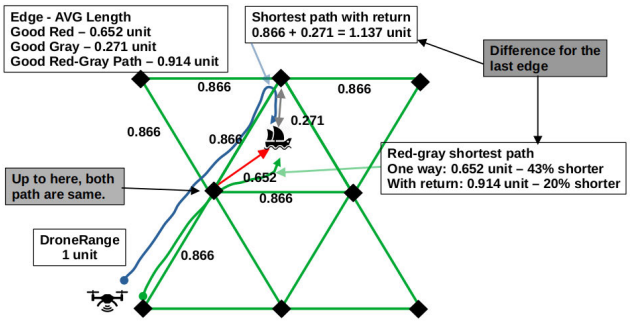
TABLE 4. Edge statistics for square grid. 643204 (4 edges * 160801 points) edges are sampled.

Edges	N	Prob.	Min	Max	Avg
gRG	199504	0.35	0.707	1	0.839
bRG	376064	0.65	1	1.366	1.075
All-RG	575568	1	0.707	1.366	0.993
gR	199504	0.45	0.502	1	0.608
bR	189448	0.14	0.502	0.999	0.749
All-R	388952	0.59	0.502	1	0.677
gG	131280	0.26	0	0.498	0.254
bG	122972	0.15	0.251	0.502	0.419
All-G	254252	0.41	0	0.502	0.334
All	643204	1.00	0	1	0.541

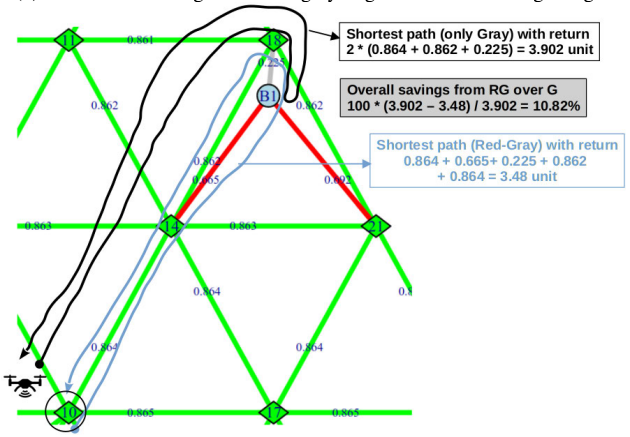
of the rescue path and present actual figures. The first column, gRG (good Red-Grey Pair), of Table 3 suggests that when we have a good red-grey edge pair, the average total length is about 0.914 units (times the Drone Range in the case of a triangular grid). This edge pair can save us from using an edge between CSs which has a distance of 0.866 unit and a grey edge which has an average length of 0.334 units. In total the length of this path is $0.866 + 0.334 = 1.2$ unit. Assuming that we use a "good grey" edge which has an average length of 0.271 units, the total length with an edge between CSs becomes $0.866 + 0.271 = 1.137$ units. So we need 1.137 units to reach the boat without using any red edge. With the red-grey edge pair, we can reach the boat in 0.652 units by using the red edge. Here the saving in reaching the boat is about 43%. If we consider the fact that we need to take the grey edge every time we take the red edge then the saving becomes about 19.3% with a 0.914 unit path. This saving is for the last edges of the path. This case is shown in Figure 16.

To find out the actual saving amount for the square grid configuration, Figure 17 shows an example scenario in which the comparison can be seen between the regular shortest path and the shortest path with the redGreySP heuristics.

The average length of the "Good Red-Grey Pair" (gRG) is listed as 0.839 units on Table 4. This edge pair can save us from using the diagonal edge between CSs which has a distance of 1 unit and a grey edge which has an average length of 0.334 units. In total the length of this path is $2 \times 0.71 + 0.334 = 1.754$ units. Assuming that we use a "good grey" edge which has an average length of 0.254 units, the total length of the path becomes $2 \times 0.71 + 0.254 = 1.674$ units. So we need 1.674 units to reach the boat without using any red edge. With the red-grey edge pair, we can reach the boat in 0.608 units by using the red edge. Here the saving in reaching the boat is about 64%. If we consider the fact that we need to take the grey edge every time we take the red edge then the saving becomes about 50% with a 0.839 unit path. This saving is for the last edges of the path. This case is shown in Figure 17.



(a) Theoretical savings from red-grey edge heuristic in triangular grid.



(b) Example savings from red-grey edge heuristic in triangular grid.

FIGURE 16. Savings from red-grey edge heuristic for Triangular CS grid configuration.

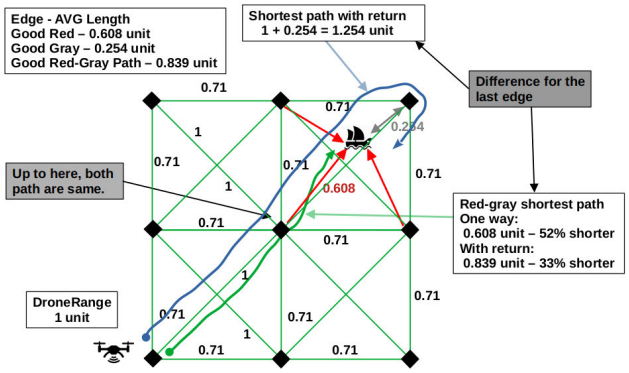
TABLE 5. Theoretical comparison of triangular and square grid CS configuration.

Grid type	AVG SP from BS	Prob. of having a Good Red-Grey Path	Prob. of using a Good Red-Grey Path	Savings 1-way	Savings Return	Mission Area per CS
Triangular	2.06	0.78	0.45	43%	20%	1.30
Square	2.07	0.82	0.31	52%	33%	0.5

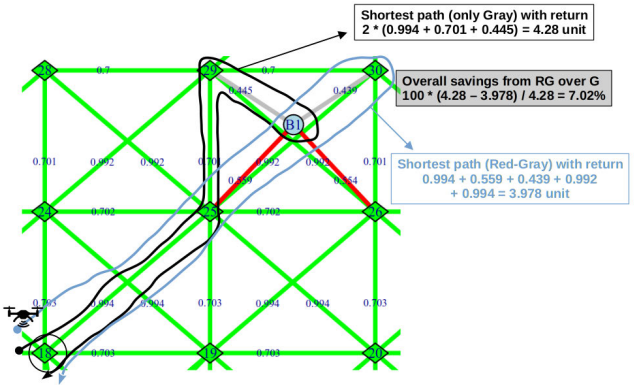
The other comparison that should be made is the coverage effectiveness of the CS grid configurations. For this, we investigated the area per CS ratios and presented analysis in Figures 18a and Figure 18b by taking the limits when the number of vertices approaches infinity.

The triangle grid limit approaches 2 triangular areas per CS while the square grid approach value of 1 square area per CS. Considering the analysis we have presented above and assuming drone range as a unit of measurement for distances, triangular grid without "blind spots" (Figure 4) gives 2 *times the area of the triangle* $= \sqrt{3} * \times 0.866^2 = 1.30\ unit^2$ per CS. On the other hand, the square grid without "blind spots"(Figure 4b) gives *Area of the square* $= 0.71^2 = 0.5\ unit^2$ per CS. This means a triangular grid can cover 2.6 times more area with the same number of CSs. In other words, a triangular grid needs fewer CSs for the same mission region.

On Table 5 we have summarized our findings for both CS deployment configurations. The "AVG SP from BS", the average length of any path from BS to any CS in the grid,

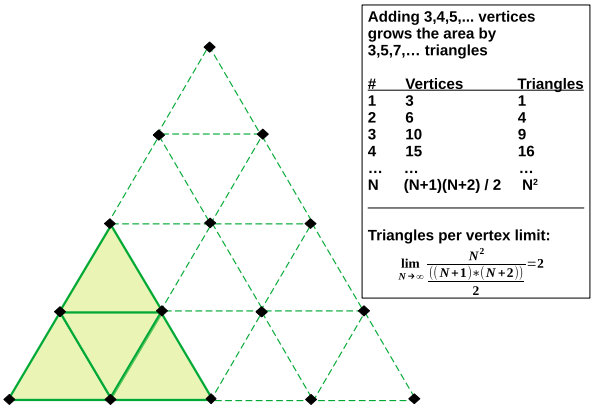


(a) Theoretical savings from red-grey edge heuristic in square grid.

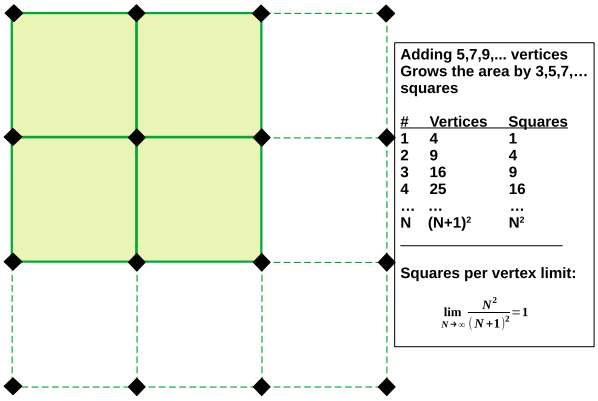


(b) Example savings from red-grey edge heuristic in square grid.

FIGURE 17. **Savings from red-grey edge heuristic for Square CS grid configuration.**



(a) The ratio of triangular area per CS for the triangular grid.



(b) The ratio of squares per CS for the square grid.

FIGURE 18. **The ratio of unit area per CS for the triangular and square grid.**

is given in drone range units. This path length is specific to the CS grid configurations shown in Figure 5. While the square grid deployment can provide a better probability of savings and a better amount of savings from the redGreySP heuristics, the number of CSs necessary for the mission is higher compared to the triangular grid. This trade-off should be considered in the CS deployment phase of the rescue mission. In Table 5 the columns **Savings 1-way** and **Savings Return** represent savings just for the last two edges of the optimised path shown in Figures 16a and in 17a.

In Algorithm 3 the pseudo-code is given for the proposed shortest path algorithm combined with the red-grey heuristic method.

The algorithm is given the optimum boat rescue permutation from the proposed TSP heuristic presented in Algorithm 2 as an input along with the graph representation of the BS, CS grid, and boats. This graph is dynamically augmented with red and grey edges according to the positions and rescue order of boats. The algorithm returns the optimum rescue tour that starts from the BS and ends on the BS after rescuing boats with the help of the CS grid. The algorithm augments the graph data structure dynamically and temporarily with the start and end boats (consecutive boats on the TSP tour) along with the incident red and grey edges when it tries to find the shortest path between these boats. As the red-grey heuristic is designed to be an ''add-on'' to any generic shortest path algorithm, this augmenting scheme ensures that the shortest path algorithm follows the order of the boats suggested by the TSP heuristic algorithm. Several functions used in the algorithm are self-explanatory. The ''E()'' and ''V()'' functions are helper functions that return edges and vertices given the structures respectively. The ''findAdjacentGoodGreyCS()'' is the function that returns the CS that can be paired with the ''red edge'' so that the total length of them will not exceed the drone range. The ''redAndgreyEdges()'' return incident red and grey edges given the vertex id of the boat. The ''asVertex()'' function creates a vertex instance given the boat id.

D. EXPERIMENTAL RESULTS AND DISCUSSIONS

We designed simulations and benchmarks to assess various performance characteristics of the proposed heuristics. In the simulations, a full rescue operation is simulated in which the drone leaves the BS, rescues boats and returns to the BS. In each simulation, the optimum rescue tour is estimated and the measurements are made by considering various performance metrics. The simulations are carried out on both triangular and square grid configurations by considering small scale datasets (less than 1000 vertices).

Algorithm 3 The Proposed Red-Grey Shortest Path Algorithm (redGreySP) in Pseudo Code

Input1: ▷ BoatPerm: The approximate TSP tour of boats, $(B_1 \dots B_N)$
Input2: ▷ Grid: Graph of BS, CSs and boats, $G(V, E)$
Output: ◁ RescueTour: The optimum rescue tour, $(V_1 \dots V_K)$
From BS to the first boat:
RescueTour ← NULL
Dynamically augment Grid with the related boats
E(Grid) ← E(Grid) + redAndgreyEdges(B_1)
V(Grid) ← V(Grid) + asVertex(B_1)
SP ← shortestPath(Grid, src=BS, dst=B_1)
if colour(lastEdge(SP)) == ''red'') **then**
    appendEdge(SP) ← E(findAdjacentGoodGreyCS(B_1))
    appendVertex(SP) ← V(findAdjacentGoodGreyCS(B_1))
end if
RescueTour ← RescueTour + SP
BoatPerm ← BoatPerm - B_1
Dynamically de-augment Grid with the related boats
This is to force the rescue order as in the BoatPerm
E(Grid) ← E(Grid) - redAndgreyEdges(B_1)
V(Grid) ← V(Grid) - asVertex(B_1)

From the first boat to the last boat:
if lastVertex(RescueTour)) == BS) **then**
    Return(RescueTour)
else
    prevBoat ← B_1
    while (BoatPerm ≠ NULL) **do**
        nextBoat ← getNext(BoatPerm)
Dynamically augment Grid with the related boats
        E(Grid) ← E(Grid) + redAndgreyEdges(prevBoat)
        V(Grid) ← V(Grid) + asVertex(prevBoat)
        E(Grid) ← E(Grid) + redAndgreyEdges(nextBoat)
        V(Grid) ← V(Grid) + asVertex(nextBoat)
        SP ← shortestPath(Grid, src=lastVertex(RescueTour), dst=nextBoat)
        if colour(lastEdge(SP)) == ''red'') **then**
            appendEdge(SP) ← E(findAdjacentGoodGreyCS(nextBoat))
            appendVertex(SP) ← V(findAdjacentGoodGreyCS(nextBoat))
        end if
        RescueTour ← RescueTour + SP
        BoatPerm ← BoatPerm - nextBoat
Dynamically de-augment Grid with the related boats
        E(Grid) ← E(Grid) - redAndgreyEdges(prevBoat)
        V(Grid) ← V(Grid) - asVertex(prevBoat)
        E(Grid) ← E(Grid) - redAndgreyEdges(nextBoat)
        V(Grid) ← V(Grid) - asVertex(nextBoat)
        prevBoat ← nextBoat
    end while
end if

From the last boat to the BS:
if lastVertex(RescueTour)) == BS) **then**
    Return(RescueTour)
else
    SP ← shortestPath(Grid, src=lastVertex(RescueTour), dst=BS)
    RescueTour ← RescueTour + SP
end if
Return(RescueTour)

These datasets consist of different numbers (20, 40, 60, 80, and 100 boats) of randomly generated boats for rescuing. For each simulation, we considered pathfinding with redGreySP heuristic and pathfinding only with a grey edge. Several standard TSP heuristics are also considered for finding the best order of boats for rescuing for each simulation. Each TSP heuristic is simulated 20 times by considering red-grey and only grey edge versions. For the simulations, we considered the real-life mission region we have presented in Section III-A in Figure 5. In the simulations, we wanted to see how much improvement the proposed ''redGreySP'' can offer for each metric we considered compared to the base case method which was pathfinding with only grey edges.

TABLE 6. Big (1000+ vertices) datasets used in the benchmarks.

Dataset	Properties	# Vertices	Link to obtain
myLattice-25x40-1000	25x40 Regular grid	1000	https://github.com/kk-1/boat-rescue
myLattice-50x40-2000	50x40 Regular grid	2000	https://github.com/kk-1/boat-rescue
myLattice-50x60-3000	50x60 Regular grid	3000	https://github.com/kk-1/boat-rescue
myRNDLattice-29x46-1000	29x46 Irregular grid (25% removal)	1000	https://github.com/kk-1/boat-rescue
myRNDLattice-58x46-2000	58x46 Irregular grid (25% removal)	2000	https://github.com/kk-1/boat-rescue
myRNDLattice-58x69-3000	58x69 Irregular grid (25% removal)	3000	https://github.com/kk-1/boat-rescue
myHexLattice-25x40-1000	25x40 Regular hex grid	1000	https://github.com/kk-1/boat-rescue
myHexLattice-50x40-2000	50x40 Regular hex grid	2000	https://github.com/kk-1/boat-rescue
myHexLattice-50x60-3000	50x60 Regular hex grid	3000	https://github.com/kk-1/boat-rescue
myRNDHexLattice-29x46-1000	29x46 Irregular hex grid (25% removal)	1000	https://github.com/kk-1/boat-rescue
myRNDHexLattice-58x46-2000	58x46 Irregular hex grid (25% removal)	2000	https://github.com/kk-1/boat-rescue
myRNDHexLattice-58x69-3000	58x69 Irregular hex grid (25% removal)	3000	https://github.com/kk-1/boat-rescue

The tentative benchmarks and simulations showed us that the selected TSP heuristics did not perform very much differently (the t-test did not show any significant statistical difference) from each other for small datasets in which about 100 or fewer vertices (boats) were used. For this reason, the proposed TSP heuristic, ''concaveTSP'', is benchmarked separately against other standard TSP heuristic algorithms using similar heuristic techniques, with bigger datasets (1000+ vertices). It performs better than the other similar heuristics as the number of vertices increases and as the regularity (grid-like geometry) of the vertex configuration increases. These aspects of the concaveTSP heuristic make it a good candidate for delivery (so many vertices) and rescue operations (sub-optimal but fast response is desired). However, the concaveTSP heuristic can be improved by considering more elaborate insertion and sub-tour (path) optimisation. While the proposed redGreySP heuristic gives savings (tour cost) independent of the selected TSP heuristic, the selection of the TSP heuristic becomes important for a big number of vertices.

In Table 6 details for these big datasets are listed.

The proposed TSP heuristic is a hybrid of the standard TSP insertion heuristic and k-Opt TSP heuristic. While the concaveTSP uses the nearest vertex insertion heuristic during the concave-hull merging stage, at the same time it performs ''k-opt'' (k = 2) type path heuristic. For this reason, we selected the ''farthest insertion'' (FI) and ''2-Opt'' heuristics. The ''farthest insertion'' heuristic is regarded as the best insertion heuristic in [35] producing better tours compared to the nearest insertion, cheapest insertion, and the nearest neighbour. The ''nearest neighbour'' (NN) heuristic was chosen with the idea that it can produce tours with the lowest AWD values. If the lesser cost edges are added to any tour earlier than the other edges, this can decrease the AWD of the tour. However, the ''nearest neighbour'' heuristic by doing this sometimes introduces risks of not being able to find lesser cost edges as it goes further and increases the AWD of the tour so much for returning to the starting vertex. We wanted to research this in the benchmarks.

The HW/SW specifications of the system that benchmarks and simulations were carried out are as follows:

- AMD Ryzen™ Threadripper™ 3960X, 24C/48T CPU
- 128 GB 3200MHz DDR4 RAM
- openSUSE Tumbleweed 64-bit Linux with kernel 5.14.6
- R version 4.1.1 (2021-08-10) - ''Kick Things''

To explain the metrics we used for the performance assessment, we tried to give a mathematical formalization of them in the following paragraphs. The TSP tour (or rescue tour)

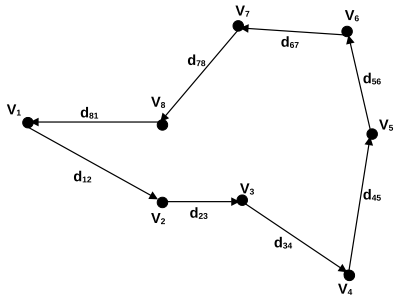


FIGURE 19. The TSP on a graph with 8 vertices.

that is presented in Figure 19, $\tau = (V_1, \ldots, V_8)$, can be an example for the metrics we explained below.

Vertices can be instances of CSs or boats. The first vertex can be regarded as the BS.

- **Tour cost:** The total ''tour cost'' is the sum of all the edge distances. The cost (Euclidean edge distance) between vertices V_i and V_j can be given as $d_{(i,j)} = Euclidean\ Distance(V_i, V_j)$. Here V_i is the starting vertex (BS in our case study) of the tour. The tour cost (cyclical) can be given with the following equation 3 below:

$$\text{Tour Cost}(\tau) = \sum_{i=1}^{N-1} d_{(i,i+1)} + d_{(N,1)} \tag{3}$$

  The TSP tour cost is rotation and direction invariant. The total cost of the tour does not change regardless of the starting vertex and the travelling direction. This cost is in the interest of the drone's party.
- **Running time:** It is the amount of CPU time consumed by the algorithm or the simulation. We measured it in seconds.
- **AWD:** Another cost that can be estimated for the TSP tour is the Average Waiting Time (AWT), which is in the interest of the boats' party. We assumed that the speed of the drone is constant over the tour. For simplicity, we ignore the ''service time'' for vertices and just consider the distance instead of time. Average total distances from the starting vertex (V_1, it is the BS) to the other vertices can be considered an alternative measure for the AWT if we want to generalize this type of cost for performance evaluation of the algorithms on the same TSP. We used AWD instead of AWT. AWD is independent of the speed and specific to the tour. However, AWD is a rotation and direction-dependent measure. The value of AWD not only depends on the starting vertex but also on the direction of the travel. Since the problem involves finding a tour, the drone should return to the BS after the rescue mission. In this sense, the ''cyclical AWD'' is estimated. The cyclical AWD of tour τ which is given in the equation 4 below:

$$\text{cyclical AWD}(\tau) = \frac{1}{N}\left(\sum_{j=2}^{N}\sum_{i=1}^{j-1} d_{(i,i+1)} + d_{(N,1)}\right) \tag{4}$$

  For the ''non-cyclical AWD'' the return to the first/starting vertex is not important. However, in some cases, the TSP algorithms (Nearest Neighbor type) use heuristics that the path they follow goes so much further away from the starting vertex that the last tour-closing edge introduces a significant cost for the overall tour cost. The ''cyclical AWD'' is such a metric that considers this penalty. On the other hand, if the only important thing is the time or distance that all the vertices are served except the starting vertex (depot), then the use of ''non-cyclical AWD'' should be considered, which is formulated in the equation 5.

$$\text{non cyclical AWD}(\tau) = \frac{1}{N-1}\left(\sum_{j=2}^{N}\sum_{i=1}^{j-1} d_{(i,i+1)}\right) \tag{5}$$

  As we said, the AWD is ''rotation and direction sensitive''. The same tour can give different AWD values depending on the starting vertex and on the direction. If the tour has long edges to the earlier vertices on its way, as these long edges will be summed over and over again for all the paths to other vertices, this type of tour will have a longer AWD value. In this sense, the ''AWD optimum'' tour should try to visit ''closer'' vertices first to get a smaller AWD value. This is similar to the ''shortest service/seek time first'' (SSTF) disk scheduling algorithm. In the opposite case, when the farthest vertex is visited first, the ''convoy effect'' increases the AWD so much. The optimum tour starting from a specific vertex does not necessarily give the min AWD. The AWD is a novel metric we proposed and measured. It is important in multi-party multi-objective optimisation problems.
- **Number of Chargings:** In the framework, the BS is also a CS. We assumed that the drone every time visits any CS charges its battery. In this sense, the number of chargings is the number of CSs drone visits on its way. This can be given by the equation 6.

$$\text{NChargings}(\tau) = \sum_{i=1}^{N} \begin{Bmatrix} 1 \text{ if } V_i \text{ is CS} \\ 0 \text{ otherwise} \end{Bmatrix} \tag{6}$$

- **Savings:** In order to see improvements that the red-grey path heuristic offers, the savings for each metric we explained above are calculated. Basically, we calculated the percentage of the improvements ($Metric_G - Metric_{RG}$) from the red-grey path heuristic ($Metric_{RG}$) over the base case of using only the grey edges ($Metric_G$). The negative savings mean loss. This can be given by the equation 7.

$$\text{Savings}\%(Metric) = \frac{100 \times (Metric_G - Metric_{RG})}{Metric_G} \tag{7}$$

The tour costs and AWD is in meters for the simulations. However, for the big dataset benchmarks, they are in

TABLE 7. Simulation results for triangular and square grid. Savings are for Red-Grey heuristics over only Grey edge usage. 20 boats randomly generated for 20 simulations. Results are in the form of mean ± standard deviation.

Grid	Metric	concaveTSP	FI	NN	2-OPT
Tri	RG Cost	452221.4 ± 45194.8	428986 ± 44761.8	472358.8 ± 44207.8	469513.4 ± 54352.4
	G Cost	548381.874 ± 51218.2	524330.3 ± 42431.6	578961.3 ± 56157.4	556431.1 ± 42991.8
	Cost Saving %	17.5 ± 3.5	18.2 ± 4.6	18 ± 8.3	15.5 ± 8.2
	RG Time	0.138 ± 0.016	0.13 ± 0.017	0.132 ± 0.021	0.128 ± 0.018
	G Time	0.115 ± 0.017	0.111 ± 0.015	0.106 ± 0.014	0.115 ± 0.025
	Time Saving %	-22.5 ± 22.6	-19 ± 24.7	-25.5 ± 25.8	-15.7 ± 26.0
	RG AWD	229580.6 ± 29556.8	212353.9 ± 27475.4	226095.7 ± 27345.6	228387.2 ± 33754.3
	G AWD	260659.2 ± 30823.6	250167.3 ± 25706.1	268733.7 ± 27410.8	264514.2 ± 24029.2
	AWD Saving %	11.4 ± 10.6	15.1 ± 7.6	15.2 ± 12.2	13.4 ± 11.6
	RG NCharging	27.3 ± 2.6	26.4 ± 2.7	28.8 ± 2.5	28.5 ± 3.2
	G NCharging	37 ± 3.0	35.6 ± 2.5	38.8 ± 3.2	37.4 ± 2.6
	NCharging Saving %	26.1 ± 4.6	26.0 ± 5.4	25.3 ± 7.4	23.7 ± 7.5
Sq	RG Cost	433795.7 ± 41318.6	413666.8 ± 39358.3	474235.7 ± 46487.9	452063.7 ± 42498.3
	G Cost	484250.594 ± 45610.3	463520.3 ± 45947.6	503022.3 ± 56043.2	487495.8 ± 44167.4
	Cost Saving %	10.4 ± 2.8	10.6 ± 5.4	5.4 ± 6.7	7.1 ± 5.6
	RG Time	0.134 ± 0.015	0.127 ± 0.012	0.135 ± 0.019	0.151 ± 0.082
	G Time	0.124 ± 0.017	0.12 ± 0.021	0.113 ± 0.015	0.109 ± 0.011
	Time Saving %	-10.5 ± 21.0	-8.0 ± 20.8	-21.2 ± 19.3	-40.0 ± 80.1
	RG AWD	212517.9 ± 24600.3	206507.0 ± 27921.9	239400.5 ± 29827.2	227244.7 ± 31700.8
	G AWD	227655.3 ± 23861.4	219690.2 ± 26102.8	232338 ± 25767.3	228410.8 ± 24335.6
	AWD Saving %	6.6 ± 6.8	5.9 ± 7.094	-4.1 ± 16.6	0.5 ± 9.6
	RG NCharging	28 ± 2.5	26.6 ± 1.9	30.4 ± 2.2	28.6 ± 2.6
	G NCharging	35.2 ± 3.1	33.8 ± 3.2	36.2 ± 3.6	35.2 ± 2.7
	NCharging Saving %	20.2 ± 3.7	21.2 ± 5.4	15.7 ± 5.3	18.7 ± 5.0

TABLE 8. Simulation results for triangular and square grid. Savings are for Red-Grey heuristics over only Grey edge usage. 40 boats randomly generated for 20 simulations. Results are in the form of mean ± standard deviation.

Grid	Metric	concaveTSP	FI	NN	2-OPT
Tri	RG Cost	720751.6 ± 45793.7	691249.3 ± 42099.4	790136.0 ± 59432.8	721864.5 ± 50161.6
	G Cost	872883.2 ± 62339.7	840797.8 ± 66965.2	932849.2 ± 80359.5	885781.1 ± 57051.8
	Cost Saving %	17.4 ± 2.3	17.6 ± 3.1	15.0 ± 6.6	18.4 ± 4.4
	RG Time	0.286 ± 0.027	0.293 ± 0.066	0.262 ± 0.027	0.284 ± 0.082
	G Time	0.233 ± 0.014	0.236 ± 0.024	0.24 ± 0.023	0.232 ± 0.021
	Time Saving %	-23.7 ± 16.6	-26.1 ± 36.2	-10.7 ± 20.8	-24.8 ± 43.4
	RG AWD	355812.2 ± 28430.5	335152.0 ± 18936.2	392548.8 ± 52073.8	353986.5 ± 25156.5
	G AWD	423383.2 ± 33027.5	409855.1 ± 32340.7	441702.0 ± 37025.5	426331.5 ± 28120.4
	AWD Saving %	15.8 ± 5	18.0 ± 4.9	11.1 ± 9.5	16.9 ± 5.0
	RG NCharging	45.6 ± 2.5	44.4 ± 2.3	49.8 ± 3.1	45.8 ± 2.0
	G NCharging	60.5 ± 3.8	58.8 ± 3.0	64.0 ± 4.5	61.4 ± 2.9
	NCharging Saving %	24.5 ± 3.2	24.4 ± 3.4	21.9 ± 6.8	25.3 ± 4.1
Sq	RG Cost	674039.5 ± 44620.1	648563.9 ± 47506.8	726830.7 ± 62542.1	672992.2 ± 41647.4
	G Cost	752414.9 ± 45426.4	723336.2 ± 41700.1	806233.0 ± 65459.6	760135.2 ± 38325.9
	Cost Saving %	10.4 ± 3.0	10.3 ± 4.2	9.7 ± 5.6	11.3 ± 6.2
	RG Time	0.283 ± 0.025	0.284 ± 0.076	0.28 ± 0.021	0.261 ± 0.016
	G Time	0.271 ± 0.083	0.244 ± 0.022	0.255 ± 0.08	0.248 ± 0.021
	Time Saving %	-9.4 ± 21.3	-17.8 ± 37.9	-15.4 ± 22.1	-6.2 ± 13.5
	RG AWD	338778.0 ± 25650.2	323931.2 ± 26247.0	366110.5 ± 44705.3	329932.0 ± 31020.0
	G AWD	364926.3 ± 24261.6	347338.3 ± 23588.8	385326.7 ± 37316.1	366898.4 ± 18717.9
	AWD Saving %	7.0 ± 5.8	6.6 ± 7.5	4.6 ± 11.7	9.9 ± 9.6
	RG NCharging	46.2 ± 1.9	45 ± 2.5	49.7 ± 3.2	46.0 ± 2.1
	G NCharging	57.7 ± 3.1	55.8 ± 2.8	60.6 ± 3.4	57.8 ± 2.8
	NCharging Saving %	19.8 ± 4.5	19.3 ± 4.9	17.9 ± 5.1	20.4 ± 5.8

TABLE 9. Simulation results for triangular and square grid. Savings are for Red-Grey heuristics over only Grey edge usage. 60 boats randomly generated for 20 simulations. Results are in the form of mean ± standard deviation.

Grid	Metric	concaveTSP	FI	NN	2-OPT
Tri	RG Cost	920434.6 ± 32296.1	901084.1 ± 39776.9	1014695.3 ± 48015.6	947449.5 ± 47647.7
	G Cost	1122678.6 ± 54088.8	1109086.4 ± 58758.0	1223358.7 ± 73658.9	1148255.4 ± 47218.4
	Cost Saving %	17.9 ± 2.3	18.7 ± 3.2	16.9 ± 4.8	17.4 ± 3.9
	RG Time	0.427 ± 0.02	0.429 ± 0.012	0.421 ± 0.019	0.431 ± 0.074
	G Time	0.4 ± 0.07	0.382 ± 0.025	0.374 ± 0.023	0.388 ± 0.068
	Time Saving %	-8.5 ± 11.8	-12.5 ± 6.5	-12.8 ± 6.8	-13.2 ± 23.4
	RG AWD	455620.9 ± 23194.3	447833.4 ± 25049.2	503701.6 ± 46465.3	464278.8 ± 31067.5
	G AWD	547482.7 ± 32740.8	539610.8 ± 30919.0	583976.3 ± 40899.3	548870.2 ± 33839.5
	AWD Saving %	16.7 ± 3.7	16.9 ± 4.8	13.3 ± 10.5	15.3 ± 5.5
	RG NCharging	62.8 ± 2.1	62.4 ± 1.8	69.2 ± 2.4	64.7 ± 2.9
	G NCharging	81.4 ± 3.2	80.6 ± 3.6	87.3 ± 4.4	82.8 ± 3.0
	NCharging Saving %	22.8 ± 3.2	22.5 ± 3.9	20.5 ± 3.6	21.8 ± 3.1
Sq	RG Cost	864874.0 ± 41174.3	852019.4 ± 39644.0	966629.2 ± 54690.3	892968.7 ± 57206.6
	G Cost	977309.5 ± 52123.8	953718.8 ± 58872.9	1052405.6 ± 61678.2	988270.3 ± 61886.8
	Cost Saving %	11.4 ± 2.4	10.5 ± 4.1	8.0 ± 5.6	9.5 ± 5.4
	RG Time	0.445 ± 0.024	0.428 ± 0.019	0.429 ± 0.021	0.456 ± 0.097
	G Time	0.427 ± 0.064	0.395 ± 0.02	0.403 ± 0.077	0.392 ± 0.022
	Time Saving %	-5.5 ± 10.2	-8.6 ± 6.2	-8.5 ± 12.6	-17.0 ± 27.1
	RG AWD	428230.9 ± 25205.0	424685.0 ± 25096.3	491356.2 ± 34682.3	445149.7 ± 30371.0
	G AWD	470800.3 ± 28575.7	463554.5 ± 31222.1	501983.1 ± 38491.1	473394.2 ± 32971.8
	AWD Saving %	8.9 ± 4.5	8.1 ± 6.3	1.8 ± 7.8	5.7 ± 7.4
	RG NCharging	63.4 ± 1.7	62.8 ± 1.7	69.8 ± 3.2	64.9 ± 2.7
	G NCharging	78 ± 2.5	75.9 ± 3.2	81.8 ± 3.8	78.0 ± 3.2
	NCharging Saving %	18.6 ± 3.5	17.2 ± 3.8	14.6 ± 4.2	16.6 ± 4.0

pixel units. The TSP heuristic used in the path-finding framework is very important. The savings can be big with the redGreySP heuristic. But if the TSP heuristic does not output a good tour in a reasonable time the savings will not be profitable to the system, as there can be a shorter tour giving overall lesser tour cost.

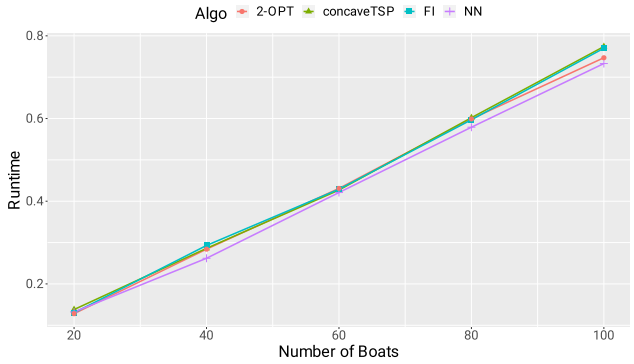
TABLE 10. Simulation results for triangular and square grid. Savings are for Red-Grey heuristics over only Grey edge usage. 80 boats randomly generated for 20 simulations. Results are in the form of mean ± standard deviation.

Grid	Metric	concaveTSP	FI	NN	2-OPT
Tri	RG Cost	1193873.4 ± 72509.5	1173711.2 ± 65385.8	1269457.7 ± 91281.2	1207666.7 ± 74836.9
	G Cost	1435808.0 ± 76420.3	1406063.3 ± 103009.1	1521348.4 ± 90490.4	1452633.4 ± 85168.4
	Cost Saving %	16.8 ± 2.3	16.4 ± 3.8	16.5 ± 3.6	16.8 ± 3.4
	RG Time	0.602 ± 0.024	0.597 ± 0.059	0.579 ± 0.016	0.599 ± 0.057
	G Time	0.545 ± 0.012	0.54 ± 0.021	0.546 ± 0.045	0.531 ± 0.013
	Time Saving %	-10.5 ± 4.1	-10.5 ± 11.4	-6.6 ± 7.6	-13.0 ± 11.4
	RG AWD	586966.9 ± 29048.0	584999.6 ± 29249.8	629485.6 ± 49728.9	599470.5 ± 41282.2
	G AWD	701345.6 ± 36936.2	684808.4 ± 53810.1	736745.9 ± 42902.6	707968.4 ± 43998.6
	AWD Saving %	16.4 ± 2.6	14.3 ± 5.4	14.4 ± 7.4	15.3 ± 3.3
	RG NCharging	83.2 ± 2.4	82.3 ± 1.7	88.7 ± 3.3	84.3 ± 2.0
	G NCharging	104.5 ± 3.7	103.0 ± 3.9	109.5 ± 4.0	105.8 ± 3.8
	NCharging Saving %	20.4 ± 2.4	20.0 ± 3.0	18.9 ± 3.1	20.2 ± 3.0
Sq	RG Cost	1087114.7 ± 41665.8	1079338.9 ± 41120.3	1202759.3 ± 62191.1	1113741.6 ± 53598.5
	G Cost	1204612.8 ± 53285.4	1202933.5 ± 45489.3	1294414.3 ± 63389.2	1251674.3 ± 75083.5
	Cost Saving %	9.7 ± 2.7	10.2 ± 2.9	6.7 ± 5.2	10.8 ± 5.0
	RG Time	0.627 ± 0.032	0.614 ± 0.02	0.624 ± 0.06	0.616 ± 0.021
	G Time	0.575 ± 0.021	0.579 ± 0.061	0.596 ± 0.077	0.563 ± 0.019
	Time Saving %	-9.1 ± 6.6	-6.8 ± 8.6	-6.2 ± 16.3	-9.6 ± 4.7
	RG AWD	540783.1 ± 27959.0	538915.9 ± 28640.4	612312.0± 46168.5	549134.7 ± 36898.8
	G AWD	589200.9 ± 28054.9	584540.6 ± 21146.8	616422.4 ± 34841.8	607477.8 ± 34821.5
	AWD Saving %	8.2 ± 3.9	7.8 ± 4.2	0.5 ± 7.8	9.4 ± 7.2
	RG NCharging	82.4 ± 2.4	82.8 ± 2.1	90.0 ± 2.9	84.1 ± 2.8
	G NCharging	97.3 ± 4.0	97 ± 4.0	102.7 ± 4.0	100.4 ± 4.9
	NCharging Saving %	15.2 ± 3.6	14.5 ± 3.4	12.3 ± 4.1	16.1 ± 4.2

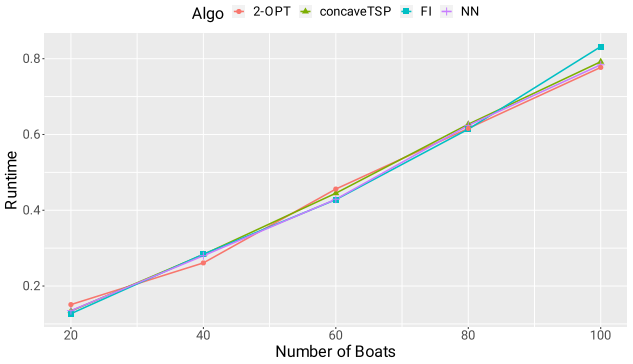
TABLE 11. Simulation results for triangular and square grid. Savings are for Red-Grey heuristics over only Grey edge usage. 100 boats randomly generated for 20 simulations. Results are in the form of mean ± standard deviation.

Grid	Metric	concaveTSP	FI	NN	2-OPT
Tri	RG Cost	1396244.1 ± 59737.2	1382759.1 ± 49286.5	1495121.1 ± 58477.8	1411860.7 ± 60752.0
	G Cost	1661578.8 ± 85371.8	1623590.9 ± 88023.8	1740390.5 ± 108055.0	1682288.9 ± 60739.8
	Cost Saving %	15.9 ± 2.2	14.7 ± 3.4	13.9 ± 4.1	16.0 ± 3.0
	RG Time	0.774 ± 0.071	0.77 ± 0.08	0.733 ± 0.014	0.747 ± 0.02
	G Time	0.721 ± 0.078	0.698 ± 0.012	0.693 ± 0.012	0.696 ± 0.011
	Time Saving %	-8.4 ± 14.7	-10.4 ± 13.0	-5.8 ± 2.3	-7.5 ± 3.2
	RG AWD	694099.1 ± 30895.1	691196.4 ± 26685.4	738222.4 ± 48601.1	714325.9 ± 42378.5
	G AWD	813660.4 ± 42993.1	792946.9 ± 47939.9	827134.7 ± 46929.7	821965.4 ± 33998.7
	AWD Saving %	14.6 ± 3.1	12.6 ± 4.5	10.5 ± 7.7	13.1 ± 4.2
	RG NCharging	101.7 ± 2.2	101.4 ± 2.0	109 ± 2.5	103.1 ± 2.6
	G NCharging	125.2 ± 4.008	1 ± 4.2	129.6 ± 4.8	126.2 ± 3.2
	NCharging Saving %	18.7 ± 2.8	17.5 ± 2.8	15.8 ± 3.3	18.3 ± 2.2
Sq	RG Cost	1323625.9 ± 47004.3	1308544.6 ± 36527.9	1433151.1 ± 72536.7	1332355.1 ± 49288.4
	G Cost	1453297.1 ± 66877.8	1436964.1 ± 63498.9	1545871.9 ± 68780.6	1475904.1 ± 47047.5
	Cost Saving %	8.9 ± 2.2	8.8 ± 3.3	7.2 ± 3.6	9.7 ± 3.5
	RG Time	0.792 ± 0.029	0.832 ± 0.13	0.784 ± 0.034	0.777 ± 0.023
	G Time	0.758 ± 0.025	0.741 ± 0.032	0.742 ± 0.028	0.74 ± 0.033
	Time Saving %	-4.6 ± 3.8	-12.6 ± 19.2	-5.8 ± 5.4	-5.1 ± 4.7
	RG AWD	658603.1 ± 24126.3	653044.7 ± 25061.3	722387.9 ± 51718.8	662079.8 ± 38094.0
	G AWD	712229.3 ± 35801.1	702810.9 ± 31125.1	738183.7 ± 31901.1	717612.3 ± 26269.5
	AWD Saving %	7.4 ± 3.2	7.0 ± 3.1	2.1 ± 6.9	7.7 ± 5.4
	RG NCharging	102.3 ± 1.5	102.0 ± 1.4	109.4 ± 3.6	103.6 ± 2.1
	G NCharging	118.1 ± 4.4	116.9 ± 4.4	123.0 ± 4.3	119.2 ± 3.6
	NCharging Saving %	13.3 ± 3.1	12.7 ± 2.8	10.9 ± 3.2	13.0 ± 2.8

The simulation results are summarized in tables below (7, 8, 9, 10, and 11) by finding the averages and standard deviations of 20 simulations. While the tables list detailed numerical results for each simulation and for each number of boats, figures (20a, 20b, 21a, 21b, 22a, 22b, 23a, and 23b) are provided to show the trend line across varying boat numbers. The standard deviations are also listed along with the averages to show the dispersion of the samples. The pairwise statistically significant differences of the averages are tested with the t-test. In tables, averages that are statistically different from all the others are marked with grey colour. The tour cost savings from the redGreySP is in the range of 10-17 % when we consider both types of CS grid configurations. This is regardless of the TSP heuristic that is utilized. The differences between triangular and square grids suggest that the tour cost savings depend on the type of grid. A triangular CS grid is better in this sense. However, the tour cost is higher in the case of the triangular CS grid. In general, the triangular CS grid provides higher savings for the red-grey heuristic over only the grey edge method. But, the tour cost, AWD, and the number of chargings are also higher. On the other hand, the triangular CS grid uses a fewer number of CSs. In Table 12 comparisons are presented for each performance

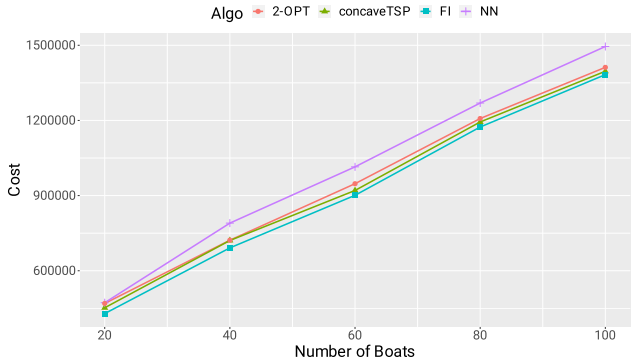


(a) AVG Runtimes for triangular grid.

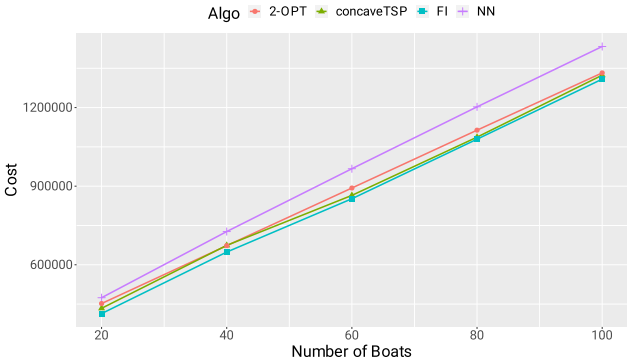


(b) AVG Runtimes for square grid.

FIGURE 20. **AVG Runtimes in seconds from 20 sims.**

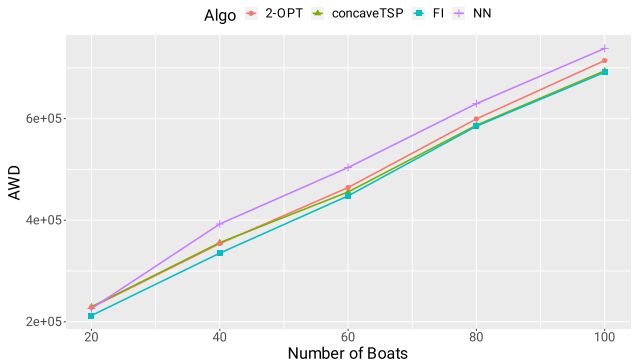


(a) AVG Tour Cost for triangular grid.

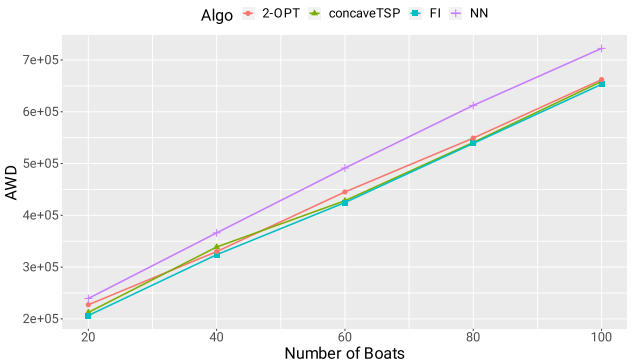


(b) AVG Tour Cost for square grid.

FIGURE 21. **AVG Tour Cost in meters from 20 sims.**



(a) AVG AWD for triangular grid.



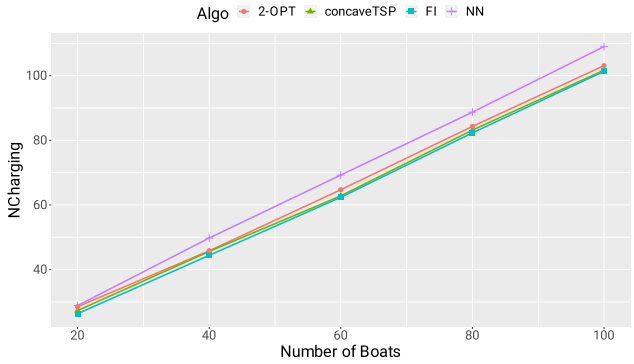
(b) AVG AWD for square grid.

FIGURE 22. **AVG AWD in meters from 20 sims.**

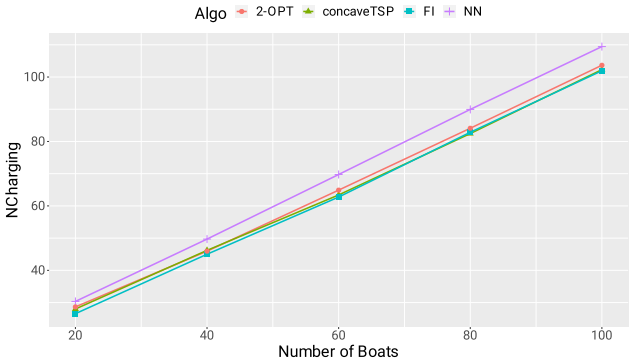
metrics according to the simulation results for different CS grid configurations.

In Figures 20a, 20b, 21a, 21b, 22a, 22b, 23a, and 23b tabular data is presented as plots for triangular and square grids. In these plots, only the algorithms that utilized the red-grey heuristic is considered. Both tables and plots suggest that all the algorithms more or less have similar performance, except for the Nearest Neighbor algorithm in some cases.

For the big dataset benchmarks, pairwise t-tests verified statistically significant differences for each metric. The best results for each metric and dataset are marked with green colour. The results from Tables 13, 14, and 15 suggest that the proposed concaveTSP heuristic is very competitive compared to other TSP heuristics. In general, it gives faster results with a close margin when we consider tour costs. Especially in a regular hexagonal grid, the proposed algorithm



(a) AVG Number of Chargings for triangular grid.



(b) AVG Number of Chargings for square grid.

FIGURE 23. AVG Number of Chargings from 20 sims.

TABLE 12. Comparison of triangular and square grid CS configuration according to simulations (Red-grey heuristic from Tables 7, 8, 9, 10, and 11).

Metric	Tri Grid	Sq Grid
Tour Cost	Higher	Lower
Tour Cost Savings%	Higher	Lower
AWD	Higher	Lower
AWD Savings%	Higher	Lower
Chargings	Not much difference	Not much difference
Chargings Savings%	Higher	Lower
Number of CSs	Lower (24)	Higher (30)

TABLE 13. Benchmark results for approximate TSP tour costs in units.

Dataset	concaveTSP	FI	NN	2-Opt
myLattice-25x40-1000	10437.0	10622.7	12225.1	10864.0
myLattice-50x40-2000	20576.3	21256.0	24264.8	21571.5
myLattice-50x60-3000	30601.7	31877.7	36396.3	32275.8
myRNDLattice-29x46-1000	13545.6	11324.5	13122.9	11597.9
myRNDLattice-58x46-2000	28001.3	22720.8	26033.6	23181.4
myRNDLattice-58x69-3000	42777.2	34129.5	38661.3	34664.3
myHexLattice-25x40-1000	10455.2	10494.8	12280.5	10730.5
myHexLattice-50x40-2000	20439.9	20534.8	21364.9	20863.0
myHexLattice-50x60-3000	30667.5	30815.9	31839.5	31193.8
myRNDHexLattice-29x46-1000	12255.9	11092.5	12798.8	11233.1
myRNDHexLattice-58x46-2000	23334.8	21695.8	24520.3	21858.7
myRNDHexLattice-58x69-3000	35494.9	32452.0	36658.8	32582.3

TABLE 14. Benchmark results for running time in seconds.

Dataset	concaveTSP	FI	NN	2-Opt
myLattice-25x40-1000	0.279	3.088	0.060	2.063
myLattice-50x40-2000	0.644	20.307	0.176	31.207
myLattice-50x60-3000	0.988	67.366	0.360	153.377
myRNDLattice-29x46-1000	0.241	3.093	0.063	2.213
myRNDLattice-58x46-2000	0.527	20.298	0.169	32.087
myRNDLattice-58x69-3000	0.858	67.522	0.365	164.172
myHexLattice-25x40-1000	0.078	3.072	0.063	2.191
myHexLattice-50x40-2000	0.117	20.378	0.189	36.102
myHexLattice-50x60-3000	0.147	68.024	0.388	160.825
myRNDHexLattice-29x46-1000	0.152	3.072	0.052	2.305
myRNDHexLattice-58x46-2000	0.273	21.087	0.182	36.376
myRNDHexLattice-58x69-3000	0.448	69.281	0.353	185.657

gives the best results in every metric we considered in the benchmarks.

TABLE 15. Benchmark results for approximate TSP tour AWD (from vertex 1) costs in units.

Dataset	concaveTSP	FI	NN	2-Opt
myLattice-25x40-1000	5300.1	5320.5	5948.3	5440.4
myLattice-50x40-2000	10395.3	10632.3	12007.0	10810.2
myLattice-50x60-3000	15389.2	15953.1	18339.8	16139.2
myRNDLattice-29x46-1000	6863.2	5668.0	6575.0	5810.1
myRNDLattice-58x46-2000	13972.7	11363.8	13010.3	11615.3
myRNDLattice-58x69-3000	21385.0	17064.1	19581.6	17348.2
myHexLattice-25x40-1000	5229.7	5254.2	6203.6	5380.4
myHexLattice-50x40-2000	10225.1	10274.0	10819.5	10469.0
myHexLattice-50x60-3000	15341.0	15415.2	16102.9	15631.5
myRNDHexLattice-29x46-1000	6257.2	5552.3	6501.7	5622.2
myRNDHexLattice-58x46-2000	11477.9	10856.9	12388.4	10930.5
myRNDHexLattice-58x69-3000	17782.4	16225.2	18310.3	16295.5

For the big datasets instead of simulations, we have just benchmarked the TSP heuristic part of the framework as we wanted to see the performance of TSP heuristics under big datasets. Here the reader should note that the redGreySP savings happen between paths from one boat to another. In this sense, for the same TSP heuristic used in the framework, the number of boats does not affect the saving percentage performance of the overall framework. This can be seen from small dataset result tables below (7, 8, 9, 10, and 11). Looking at the "Cost saving" rows this claim can be verified. In Tables 13, 14, and 15 custom datasets are used to benchmark algorithms with big number of vertices (1000+). In delivery operations, the number of vertices can be much bigger than the rescue operations. These datasets although do not represent real-life delivery configurations can be useful for comparison purposes. We had two goals for creating these custom datasets. The first goal was to see how the proposed TSP heuristics perform with bigger datasets compared to other heuristics since for small datasets there were no big statistically significant differences. The second goal was based on our observation during trial simulations that the proposed TSP heuristic performed better then the other heuristics in the regular geometric layout of the vertices. To test these we created regular and hexagonal lattice layouts with varying sizes in TSPLIB format. These lattices are also

deformed by randomly removing 25% of the vertices from the regular layout. The datasets and the detailed results can be downloaded from https://github.com/kk-1/boat-rescue.

IV. CONCLUSION AND FUTURE WORKS

We tried to address a very specific instance of a novel problem of optimised pathfinding in general drone-based operations assisted with an optimised CS grid. Namely, we considered the ''single drone-multiple entities'' case, which is an instance of the classical TSP. We tried to highlight the synergy between the regularity of the proposed optimised CS grid and the proposed heuristics.

The optimised CS grid provides complete coverage of the mission region with the min number of CSs and without any blind-spot. In Section III-D we listed the trade-offs associated with the CS grid configurations we studied.

The proposed redGreySP heuristic depends on the CS grid configuration and provides savings for the tour cost.

The proposed concaveTSP heuristic is not the best TSP heuristic but it is a fast approximation heuristic. It offered a good synergy between insertion heuristic and 2-Opt like sub-tour improvement heuristic.

The novelties and contributions of our current work can be listed as follows:

- Proposal of a novel framework that consists of the generic region coverage method with CSs and the optimum pathfinding for the entities in the covered region.
- Proposal of novel CS grid configurations for optimum coverage of the mission region with the minimum number of CSs and without ''blind spots'' (points in the mission region where the drone can not reach).
- Proposal of a new and flexible geometry-based paradigm, ''concaveTSP'', that can be integrated with various existing and new heuristics for the TSP heuristic algorithms.
- Proposal of a novel add-on type heuristic, ''redGreySP'', for a generic shortest path algorithm that exploits the geometric regularities of the proposed CS grid.

Currently, the redGreySP heuristic is designed as an add-on for any generic shortest path algorithm. In this sense, there can be some trade-offs. As future work, we can suggest integrating it fully and creating a special shortest path algorithm. This algorithm should work with dynamic data structures.

The ''jumps'' from one boat to another are not included in our study for simplicity. For this, we considered augmenting the graph data structures with ''yellow'' edges to represent possible jumps among boats that are very close to each other. This may happen in the regions bounded by multiple CSs. However, this scheme introduces another $\mathcal{NP}$-Hard problem, namely ''bin-packing''. The algorithm should see the yellow edge distances as ''weights'' and should try to fit them into ''bins'' as large as ''drone range''. The min number of ''bins'' should be found for an energy-optimised path.

Multiple drones from single/multiple BS cases can be studied. They are special Vehicular Routing Problem (VRP) cases. For the multiple BS case, we can offer the Voronoi Tessellation method used in the study [43] for dividing the large mission region into smaller regions based on the BS positions. By using this division scheme the drone-based delivery or rescue operations can be done in parallel over each sub-region.

REFERENCES

[1] H. Shakhatreh, A. H. Sawalmeh, A. Al-Fuqaha, Z. Dou, E. Almaita, I. Khalil, N. S. Othman, A. Khreishah, and M. Guizani, ''Unmanned aerial vehicles (UAVs): A survey on civil applications and key research challenges,'' *IEEE Access*, vol. 7, pp. 48572–48634, 2019, doi: 10.1109/ACCESS.2019.2909530.

[2] Dronelife. (2021). *A Drone Battery That Charges in 5 Minutes*. Dronlife News. Accessed: Jan. 4, 2021. [Online]. Available: https://dronelife.com/2020/08/01/a-drone-battery-that-charges-in-5-minutes/

[3] K. I. Kilic and L. Mostarda, ''Optimum path finding framework for drone assisted boat rescue missions,'' in *Advanced Information Networking and Applications*, L. Barolli, I. Woungang, and T. Enokido, Eds. Cham, Switzerland: Springer, 2021, pp. 219–231, doi: 10.1007/978-3-030-75078-7_23.

[4] R. M. Karp, ''Reducibility among combinatorial problems,'' in *Proc. Symp. Complex. Comput. Comput. Complex. Comput. Comput.*, R. E. Miller, J. W. Thatcher, and J. D. Bohlinger, Eds. Boston, MA, USA: Springer, Mar. 1972, pp. 85–103, doi: 10.1007/978-1-4684-2001-2_9.

[5] S. Khuller, A. Malekian, and J. Mestre, ''To fill or not to fill: The gas station problem,'' *ACM Trans. Algorithms*, vol. 7, no. 3, pp. 1–16, Jul. 2011, doi: 10.1145/1978782.1978791.

[6] Marinei. (2020). *The Growing Role for Aerial Drones in the Maritime Industry*. Marinei News. Accessed: Dec. 29, 2020. [Online]. Available: https://www.marine-i.co.U.K./news/article/80/the-growing-role-for-aerial-drones-in-the-maritime-industry

[7] M. Silvagni, A. Tonoli, E. Zenerino, and M. Chiaberge, ''Multipurpose UAV for search and rescue operations in mountain avalanche events,'' *Geomatics, Natural Hazards Risk*, vol. 8, no. 1, pp. 18–33, Jan. 2017, doi: 10.1080/19475705.2016.1238852.

[8] Y. Karaca, M. Cicek, O. Tatli, A. Sahin, S. Pasli, M. F. Beser, and S. Turedi, ''The potential use of unmanned aircraft systems (drones) in mountain search and rescue operations,'' *Amer. J. Emergency Med.*, vol. 36, no. 4, pp. 583–588, Apr. 2018, doi: 10.1016/j.ajem.2017.09.025.

[9] J. N. McRae, C. J. Gay, B. M. Nielsen, and A. P. Hunt, ''Using an unmanned aircraft system (Drone) to conduct a complex high altitude search and rescue operation: A case study,'' *Wilderness Environ. Med.*, vol. 30, no. 3, pp. 287–290, Sep. 2019, doi: 10.1016/j.wem.2019.03.004.

[10] B. Mishra, D. Garg, P. Narang, and V. Mishra, ''Drone-surveillance for search and rescue in natural disaster,'' *Comput. Commun.*, vol. 156, pp. 1–10, Apr. 2020, doi: 10.1016/j.comcom.2020.03.012.

[11] A. Claesson, S. Schierbeck, J. Hollenberg, S. Forsberg, P. Nordberg, M. Ringh, M. Olausson, A. Jansson, and A. Nord, ''The use of drones and a machine-learning model for recognition of simulated drowning victims—A feasibility study,'' *Resuscitation*, vol. 156, pp. 196–201, Nov. 2020, doi: 10.1016/j.resuscitation.2020.09.022.

[12] M. Hassanalian and A. Abdelkefi, ''Classifications, applications, and design challenges of drones: A review,'' *Prog. Aerosp. Sci.*, vol. 91, pp. 99–131, May 2017, doi: 10.1016/j.paerosci.2017.04.003.

[13] K. Kuru, ''Planning the future of smart cities with swarms of fully autonomous unmanned aerial vehicles using a novel framework,'' *IEEE Access*, vol. 9, pp. 6571–6595, 2021, doi: 10.1109/ACCESS.2020.3049094.

[14] B. Galkin, J. Kibilda, and L. A. DaSilva, ''UAVs as mobile infrastructure: Addressing battery lifetime,'' *IEEE Commun. Mag.*, vol. 57, no. 6, pp. 132–137, Jun. 2019, doi: 10.1109/MCOM.2019.1800545.

[15] C. H. Choi, H. J. Jang, S. G. Lim, H. C. Lim, S. H. Cho, and I. Gaponov, ''Automatic wireless drone charging station creating essential environment for continuous drone operation,'' in *Proc. Int. Conf. Control, Autom. Inf. Sci. (ICCAIS)*, Oct. 2016, pp. 132–136, doi: 10.1109/iccais.2016.7822448.

[16] S. Yin, Y. Zhao, and L. Li, ''Resource allocation and basestation placement in cellular networks with wireless powered UAVs,'' *IEEE Trans. Veh. Technol.*, vol. 68, no. 1, pp. 1050–1055, Jan. 2019, doi: 10.1109/TVT.2018.2883093.

[17] J. Ouyang, Y. Che, J. Xu, and K. Wu, "Throughput maximization for laser-powered UAV wireless communication systems," in *Proc. IEEE Int. Conf. Commun. Workshops (ICC Workshops)*, May 2018, pp. 1–6, doi: 10.1109/ICCW.2018.8403572.

[18] B. Michini, T. Toksoz, J. Redding, M. Michini, J. How, M. Vavrina, and J. Vian, "Automated battery swap and recharge to enable persistent UAV missions," in *Proc. Infotech Aerospace*, Mar. 2011, p. 1405, doi: 10.2514/6.2011-1405.

[19] H. Huang and A. V. Savkin, "A method of optimized deployment of charging stations for drone delivery," *IEEE Trans. Transport. Electrific.*, vol. 6, no. 2, pp. 510–518, Jun. 2020, doi: 10.1109/TTE.2020.2988149.

[20] H. Huang and A. V. Savkin, "Optimal deployment of charging stations for aerial surveillance by UAVs with the assistance of public transportation vehicles," *Sensors*, vol. 21, no. 16, p. 5320, Aug. 2021, doi: 10.3390/s21165320.

[21] K. Sundar and S. Rathinam, "Algorithms for routing an unmanned aerial vehicle in the presence of refueling depots," *IEEE Trans. Autom. Sci. Eng.*, vol. 11, no. 1, pp. 287–294, Jan. 2014, doi: 10.1109/TASE.2013.2279544.

[22] P. Maini and P. B. Sujit, "On cooperation between a fuel constrained UAV and a refueling UGV for large scale mapping applications," in *Proc. Int. Conf. Unmanned Aircr. Syst. (ICUAS)*, Jun. 2015, pp. 1370–1377, doi: 10.1109/ICUAS.2015.7152432.

[23] C. C. Murray and A. G. Chu, "The flying sidekick traveling salesman problem: Optimization of drone-assisted parcel delivery," *Transp. Res. C, Emerg. Technol.*, vol. 54, pp. 86–109, May 2015, doi: 10.1016/j.trc.2015.03.005.

[24] R. G. Mbiadou Saleu, L. Deroussi, D. Feillet, N. Grangeon, and A. Quilliot, "An iterative two-step heuristic for the parallel drone scheduling traveling salesman problem," *Networks*, vol. 72, no. 4, pp. 459–474, Dec. 2018, doi: 10.1002/net.21846.

[25] I. Khoufi, A. Laouiti, and C. Adjih, "A survey of recent extended variants of the traveling salesman and vehicle routing problems for unmanned aerial vehicles," *Drones*, vol. 3, no. 3, p. 66, Aug. 2019, doi: 10.3390/drones3030066.

[26] R. G. M. Saleu, L. Deroussi, D. Feillet, N. Grangeon, and A. Quilliot, "The parallel drone scheduling problem with multiple drones and vehicles," *Eur. J. Oper. Res.*, Sep. 2021, doi: 10.1016/j.ejor.2021.08.014.

[27] R. Roberti and M. Ruthmair, "Exact methods for the traveling salesman problem with drone," *Transp. Sci.*, vol. 55, no. 2, pp. 315–335, Mar. 2021, doi: 10.1287/trsc.2020.1017.

[28] S. A. Vásquez, G. Angulo, and M. A. Klapp, "An exact solution method for the TSP with drone based on decomposition," *Comput. Oper. Res.*, vol. 127, Mar. 2021, Art. no. 105127, doi: 10.1016/j.cor.2020.105127.

[29] T. Lust and J. Teghem, "The multiobjective traveling salesman problem: A survey and a new approach," in *Advances in Multi-Objective Nature Inspired Computing*, C. A. C. Coello, C. Dhaenens, and L. Jourdan, Eds. Berlin, Germany: Springer, 2010, pp. 119–141, doi: 10.1007/978-3-642-11218-8_6.

[30] S. Lianshuan and L. Zengyan, "An improved Pareto genetic algorithm for multi-objective TSP," in *Proc. 5th Int. Conf. Natural Comput.*, vol. 4, 2009, pp. 585–588, doi: 10.1109/ICNC.2009.510.

[31] X. Guo, M. Ji, Z. Zhao, D. Wen, and W. Zhang, "Global path planning and multi-objective path control for unmanned surface vehicle based on modified particle swarm optimization (PSO) algorithm," *Ocean Eng.*, vol. 216, Nov. 2020, Art. no. 107693, doi: 10.1016/j.oceaneng.2020.107693.

[32] M. M. Flood, "The traveling-salesman problem," *Oper. Res.*, vol. 4, no. 1, pp. 61–75, 1956. [Online]. Available: http://www.jstor.org/stable/167517

[33] G. A. Croes, "A method for solving traveling-salesman problems," *Oper. Res.*, vol. 6, no. 6, pp. 791–812, Dec. 1958. [Online]. Available: http://www.jstor.org/stable/167074

[34] M. Jünger, G. Reinelt, and G. Rinaldi, "The traveling salesman problem," in *Network Models* (Handbooks in Operations Research and Management Science), vol. 7, M. O. Ball, T. L. Magnanti, C. L. Monma, and G. L. Nemhauser, Eds. Amsterdam, The Netherlands: Elsevier, 1995, ch. 4, pp. 225–330, doi: 10.1016/S0927-0507(05)80121-5.

[35] D. J. Rosenkrantz, R. E. Stearns, and P. M. Lewis, "An analysis of several heuristics for the traveling salesman problem," in *Fundamental Problems in Computing: Essays in Honor of Professor Daniel J. Rosenkrantz*, S. S. Ravi and S. K. Shukla, Eds. Dordrecht, The Netherlands: Springer, 2009, pp. 45–69, doi: 10.1007/978-1-4020-9688-4_3.

[36] A. Galton and M. Duckham, "What is the region occupied by a set of points?" in *Geographic Information Science*, M. Raubal, H. J. Miller, A. U. Frank, and M. F. Goodchild, Eds. Berlin, Germany: Springer, 2006, pp. 81–98, doi: 10.1007/11863939_6.

[37] M. Duckham, L. Kulik, M. Worboys, and A. Galton, "Efficient generation of simple polygons for characterizing the shape of a set of points in the plane," *Pattern Recognit.*, vol. 41, no. 10, pp. 3224–3236, Oct. 2008, doi: 10.1016/j.patcog.2008.03.023.

[38] V. G. Deineko and G. J. Woeginger, "The convex-hull-and-k-line travelling salesman problem," *Inf. Process. Lett.*, vol. 59, no. 6, pp. 295–301, 1996, doi: 10.1016/0020-0190(96)00125-1.

[39] J. Jones and A. Adamatzky, "Computation of the travelling salesman problem by a shrinking blob," *Natural Comput.*, vol. 13, no. 1, pp. 1–16, Mar. 2014, doi: 10.1007/s11047-013-9401-x.

[40] A. Moreira and M.-Y. Santos, "Concave hull: A k-nearest neighbours approach for the computation of the region occupied by a set of points," in *Proc. Int. Conf. Comput. Graph. Theory Appl. (GRAPP)*, 2007, pp. 61–68.

[41] J. Gombin, R. Vaidyanathan, and V. Agafonkin. (2020). *Concaveman: A Very Fast 2D Concave Hull Algorithm, R Package Version 1.1.0*. [Online]. Available: https://CRAN.R-project.org/package=concaveman

[42] J.-S. Park and S.-J. Oh, "A new concave hull algorithm and concaveness measure for n-dimensional datasets," *J. Inf. Sci. Eng.*, vol. 29, no. 2, pp. 379–392, 2013.

[43] K. I. Kilic, O. Gemikonakli, and L. Mostarda, "Voronoi tesselation-based load-balanced multi-objective priority-based heuristic optimisation for multi-cell region coverage with UAVs," *Int. J. Web Grid Services*, vol. 17, no. 2, pp. 152–178, 2021, doi: 10.1504/IJWGS.2021.114574.

KEMAL IHSAN KILIC received the M.Sc. degrees in computer science from the University of Maryland, College Park, in 1998, and in sustainable environment and energy systems from Middle East Technical University, North Cyprus Campus, in 2018. He is currently pursuing the Ph.D. degree with the Computer Science Division, University of Camerino. His research interests include computer vision, image processing, machine learning, and wireless sensor networks.

LEONARDO MOSTARDA (Member, IEEE) received the Ph.D. degree from the Computer Science Department, University of L'Aquila, in 2006. He cooperated with the European Space Agency (ESA) on the CUSPIS FP6 Project to design and implement novel security protocols and secure geo tags. In 2007, he was a Research Associate with the Distributed System and Policy Group, Department of Computing, Imperial College London, where he was working on the UBIVAL EPRC Project in cooperation with Cambridge, Oxford, Birmingham, and UCL for building a novel middleware to support the programming of body sensor networks. In 2010, he was a Senior Lecturer with the Department of Distributed Systems and Networking, Middlesex University. He is currently an Associate Professor with Department of Computer Science, Camerino University, Italy, and the CEO of Bilancio CO_2 Zero. His main research activities include the area of wireless sensor networks, middleware, security, and various aspect of distributed systems.