UNIVERSITÀ DEGLI STUDI DI CAMERINO

SCHOOL OF ADVANCED STUDIES

DOTTORATO DI RICERCA IN SCIENZE E TECNOLOGIE
COMPUTER SCIENCE - XXXIV CICLO

Geometry-Based Optimization Heuristics for Region Coverage and Pathfinding in Drone-Based Operations

Relatore
Dr. Leonardo Mostarda

Dottorando
Kemal Kemal

Commissione Esaminatrice
Jury1 name
Jury2 name

A.A. 2021/2022

UNIVERSITY OF CAMERINO

SCHOOL OF ADVANCED STUDIES

DOCTOR OF PHILOSOPHY IN SCIENCES AND TECHNOLOGY
COMPUTER SCIENCE - XXXIV CYCLE

Geometry-Based Optimization Heuristics for Region Coverage and Pathfinding in Drone-Based Operations

Supervisor
Dr. Leonardo Mostarda

PhD Candidate
Kemal Kemal

Doctoral Examination Committee
Jury1 name
Jury2 name

A.A. 2021/2022

“In loving memory of my grandfather Kani.”

Abstract

EVs (Electric Vehicles), especially EFVs (Electric Flying Vehicles) are important enabling technologies for sustainable living on our planet. EVs have a smaller carbon footprint than gasoline cars, even including charging. However, for more effective use of EVs, the limited onboard energy of EVs needs novel research. In principle, the thesis addresses this issue and proposes geometry-based optimization heuristics for making drone-based operations more effective. Drones are very versatile vehicles that can be utilized in many critical operations. They can be mobile IoT stations, providing mission-critical data. Drones can fly directly from one point to another following the shortest possible path. Two important operation types are studied in the thesis. Namely, region coverage and pathfinding. While the region coverage operations involve the static configuration of drones, pathfinding involves the dynamic utilization of drones. For both types of operations, geometry-based optimization heuristics are proposed and benchmarked. The improvements over the base case methods are measured in an experimental setup. Statistical meanings of the results are discussed and the most effective methods are stated. While the energy (or flight distance to reach the goal) was the main objective for the optimizations, "scenario-based weighted multi-objective scoring" is utilized in the proposed optimization frameworks. For the region coverage, which is a form of the SCP (Set Cover Problem), the thesis proposes a novel multi-objective priority-based heuristic optimization framework, in which EAs (Evolutionary Algorithms) like GA (Genetic Algorithm), GenSA (Generalised Simulated Annealing), and DEoptim (Differential Evolution Optimization) are utilized to "score" the drone configuration. The SCP is one of the $\mathcal{NP}$-Hard problems. EAs provide easy problem formulation and approximate the optimum solution effectively. GA and GenSA are supplied initial solutions from a fast hexagonal circle-packing algorithm by considering homogeneous drone altitudes. In some cases, this helped slow algorithms like GenSA in finding the best coverage. GA did not benefit very much from the supplied initial solution. However, without an initial solution GA was the best algorithm for finding drone configurations with minimum total drone distance. DEoptim was the quickest algorithm, although the implementation utilized in the experiments did not accept an initial solution. For the pathfinding, the thesis proposes a novel optimization framework that consists of the optimized Charging Station (CS) grid and the pathfinding heuristics for the drone. Novelties include a custom TSP (Travelling Salesman Problem) approximation heuristic (concaveTSP) and a custom SP (Shortest Path) algorithm called "redGraySP". These heuristics are assessed for two different (triangular and square) CS grid configurations that are optimized for the drone range. The case study of a boat rescue operation that is carried out in the sea is presented. The minimization of the "flight distance" and "number of chargings" are the objectives for the drone party and the minimization of the "average waiting distance" (AWD) is the objective for the boat party. The "single drone with many entities" case which is a form of TSP is studied. Mathematical analysis and simulation results for the effectiveness of the pathfinding heuristic are presented. The performance of the novel and fast custom TSP heuristic was assessed. The tour cost savings (over the base case) from the redGraySP is in the range of 10-17 % when we consider both types of CS grid configurations.

Acknowledgements

Planted in the house of the LORD, they shall flourish in the courts of our God.
They shall bear fruit even in old age, they will stay fresh and green,
to proclaim: "The LORD is just; my rock, in whom there is no wrong."

Psalm 92: 14-16

Infinite thanks to our infinitely good God. As the Psalm goes above, he sustained me in my "advanced years" to complete my studies.

Lots of thanks to my wife Yücel for her love, patience, and encouragement. I am thankful to my family for their support as well. I am very grateful to my adviser Dr. Leonardo Mostarda for his help. I have to extend my thanks to Dr. Ali Cevat Taşıran, Dr. Murat Fahrioğlu, and Dr. Orhan Gemikonaklı too for their mentoring and support. Many thanks to all my teachers, as my thesis is the fruit of the works of their hands.

I want to thank the University of Camerino for giving me chance to carry out my studies. I will not forget the welcoming hospitality of the city of Camerino as well.

Contents

List of Algorithms

List of Figures

List of Tables

1. Introduction

⟨chap:intro⟩

"Begin at the beginning," the King said, very gravely,
"and go on till you come to the end: then stop."

"Alice in Wonderland" by Lewis Carroll

EVs are promising technology and essential to sustainability efforts. Not only do they offer economical savings [44, 19], but also they have a low carbon footprint [49, 37] compared to GVs (Gas Vehicles). Drones being EFVs, they also inherit these attributes. They are versatile "mobile IoT (Internet of Things)" platforms. For small drones, their size can be advantageous as they can be carried easily and they can fly around obstacles easily. Table 1.0.1 gives a short list of pros and cons of drones[1].

Table 1.0.1: Pros and cons of drones as "flying IoT".

⟨tab:pro-con⟩

Pros	Cons
Straight flight over obstacles	Limited onboard energy
Precision (GPS)	Little wind can effect flight
Easy deployment	Uncertain regulations
Autonomous mission	Small size can be problem
Quality aerial imaging	Privacy problems
Small size can be useful	
Can carry diverse types of sensors	

However, because of their small size, the amount of battery (energy) they can carry is also small. In this sense, energy optimization is vital to increase operational effectiveness. On the other hand, "ground traffic" has many when we consider the land use for road construction, accidents, and high carbon footprint. The use of EFVs can alleviate these problems. But without smart optimized "charging" infrastructure and smart pathfinding over this infrastructure EFVs can not be used effectively for a greater range of transportation.

In the light of these motivations, the thesis addressed the issue of limited onboard energy of Electric Vehicles. Specifically, multi-party (drones, boats), multi-objective (energy, waiting time) optimization frameworks (coverage/pathfinding) in which novel geometry-based heuristics are utilized for drone-based operations are proposed in the thesis. While the proposed geometry-based heuristics are presented in the case studies, they can also be used in any coverage and pathfinding contexts.

[1] Partly from: https://onlinemasters.ohio.edu/blog/the-pros-and-cons-of-unmanned-aerial-vehicles-uavs/

Two fundamental drone-based operations are considered in the thesis. The region coverage is one of the basic operations requiring "static" deployment of the drones. On the other hand, the rescue pathfinding case study represents the "dynamic" use of drones. The proposed frameworks are benchmarked in case studies like disaster region coverage and pathfinding for sea rescue operations. Figure 1.0.1 summarizes the research milestones accomplished for drone-based operations in the thesis. The poster on page 121 in Appendix A summarizes the research carried out for the thesis.

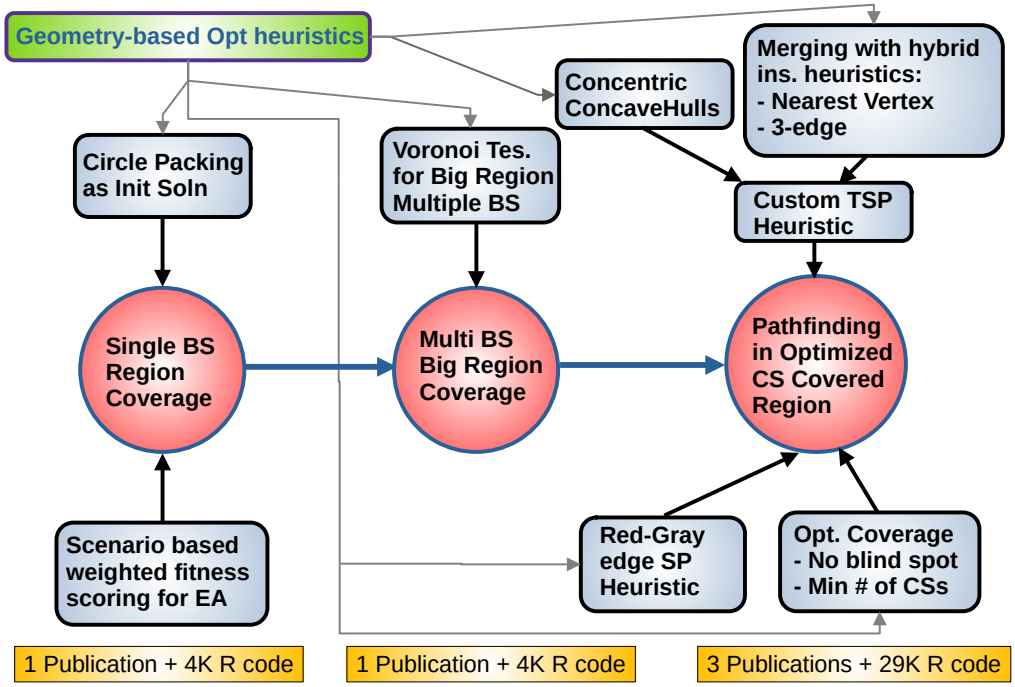


Figure 1.0.1: The summary of the research done for the thesis.

⟨fig:rsum⟩

The region coverage work is considered in the context of natural disaster situations or temporal events. In the case of natural disasters, the serving BS (Base Station) can be affected requiring vital and urgent ad hoc connection to the afflicted region. For temporal events like sports or concerts, the service region can be out of the reach of the current communication infrastructure. In such cases, drones are versatile and economic for extending the existing communication network. For such case studies, single BS and multi BS frameworks are proposed and studied in the thesis. The region coverage problem is an $\mathcal{NP}$-Hard problem and is related to the SCP, which was one of the 21 $\mathcal{NP}$-Complete problems listed by Karp in [79]. In fact, the region coverage is an instance of a geometric version of the SCP. The SCP (minimum) ILP (Integer Linear Program) formulation is given in equation 1.0.1 below [140, p. 108-109]:

$$\begin{aligned} &\text{minimize} \quad \sum_{s \in \mathcal{S}} x_s \text{ (minimize the number of sets)} \\ &\text{subject to} \quad \sum_{s:\, e \in s} x_s \geqslant 1 \text{ for all } e \in \mathcal{U} \text{ (cover every element of the universe)} \\ &\qquad x_s \in \{0,1\} \text{ for all } s \in \mathcal{S} \text{ (every set is either in the set cover or not)} \end{aligned} \tag{1.0.1}$$

?⟨eq:scp⟩?

The solution space is big and the formulation for the optimization is difficult (Figure **??**). To overcome these difficulties the thesis proposed the use of EAs in a multi-objective optimization

framework. The EAs perform solution space search in a heuristic way to find the global maximum/minimum. The proposed framework offers flexibility to adjust the priorities associated with the objectives according to the scenario in which optimization is necessary.

The pathfinding work is presented in the context of a "boat rescue" case study which was an aborted project once considered for the sea of Marche in Italy. A statistical data according to study in [54] states:

> Every year, boating activities require to deal with a high number of rescuing calls. In Italy in 2013, 6166 boats called for help. During the summer period (35 days) about 1152 people called for aid in Marche region. . . . false alarms. More precisely, only 2.4% of the calls requires rescuing, thus lifeboat assistance.

This means roughly a rescue call every other day. For a such frequency of calls, it can be more economical to use small drones instead of sending a helicopter or a big rescue boat when we consider the energy and maintenance costs. In addition to this observation, the boat rescue case study is a good presentation tool for explaining the theory of the proposed pathfinding heuristics and the optimization framework which is designed for general operations involving EVs. The proposed framework consists of two main elements. Namely, the optimized CS Grid and the pathfinding heuristics. The synergy between the CS Grid and the pathfinding heuristics is established and exploited. While the CS Grid is optimized for maximum region coverage and no "blind spots" (regions out of reach of the drone), the pathfinding heuristics are optimized with a weight-based scheme by considering the drone energy and the boat waiting time. The pathfinding heuristics consist of two fundamental algorithms. Namely, the custom-designed TSP algorithm called "concaveTSP" and the customized SP algorithm called "redGraySP". While the concaveTSP finds the "optimum rescue order" of the boats, the proposed "redGraySP" finds the shortest rescue paths between boats over the CS Grid "jumps". This optimization framework has multi-party and multi-objective nature:

- **The CS Grid:** Optimized for **region coverage** (main objective) through geometry-based heuristics for the **mission region**.
- **The Pathfinding Algorithm:** Optimized for **energy** (main objective) through geometry-based heuristics for the **mission drones**.

The TSP ILP (Integer Linear Program) formulation is given in equation 1.0.2 below [139, p. 347-348]:

$$
\begin{aligned}
& c_{ij} \text{ cost of using edge from city } i \text{ to } j \\
x_{ij} \;=\; & \begin{cases} 1 & \text{if city } j \text{ is visited immediately after city } i \\ 0 & \text{otherwise} \end{cases} \\
\text{minimize} \quad & \sum_i \sum_j c_{ij} x_{ij} \\
\text{subject to} \quad & \sum_{j=1}^{n} x_{ij} = 1, \;\; i = 0,1,\ldots,n-1 \quad \text{(go-to constraints)} \\
& \sum_{i=1}^{n} x_{ij} = 1, \;\; j = 0,1,\ldots,n-1 \quad \text{(come-from constraints)} \\
& t_j \;\geq\; t_i + 1 - n(1-x_{ij}), \quad i \geq 0,\; j \geq 1,\; i \neq j \quad \text{(subtour elim. constraints)} \quad (1.0.2)
\end{aligned}
$$

?{eq:tsp}?

The "subtour elimination constraints" [139, p. 347-348] in equation 1.0.2 can be explained in

long formulation below in equation 1.0.3:

s_i denotes the i^{th} city visited

TSP tour of n cities $s_0 = 0, s_1, s_2, \ldots, s_{n-1}$

t_i denotes the number of the stop along the tour at which city i is visited

For example: $t_0 = 0,\ t_{s_1} = 1,\ t_{s_2} = 2,$ etc...

In general, $t_{s_i} = i,\ \ i = 0, 1, \ldots, n-1$

$$t_j \ = \ t_i + 1, \text{ if } x_{ij} = 1,\ t_i \text{ an integer } \in [0..(n-1)]$$

$$t_j \ \geq \ \begin{cases} t_i + 1 - n & \text{if } x_{ij} = 0 \\ t_i + 1 & \text{if } x_{ij} = 1 \end{cases} \qquad (1.0.3)$$?{eq:sub-tour-elim}

In principle, the proposed heuristics either exploit the existing geometric configurations of the entities or introduce geometric regularities for their configurations. Fundamental geometric structures and techniques used in the thesis can be listed as follows:

- **Voronoi-Delaunay Structures:** These geometric structures partition big regions into smaller "cells" given the "generating points" [13, 33, 34]. The partitioning is such that every point in the cell is always closest to the associated generating point. These structures are good for "Neighborhood" discovery. The division of the region according to the "potential" of the generating points is also useful to divide the work of the big region into smaller manageable "cells". In that sense, they have useful global and local properties. In the region coverage, Voronoi Tessellation is used to divide the big mission region into smaller regions given the nearby BSs. Delaunay Triangulation is basically formed by joining the generating points and forming triangles according to special rules. The edges of the Delaunay Triangulation can be useful to find the neighbor of the generating points [116]. An example for these geometric Structures can be seen in Figure 1.0.2. In Figure 1.0.2a the mission region is divided among the available BSs by using Voronoi Tessellation. In Figure 1.0.2b the BSs that are neighbors of each other can be found by following the adjacent vertices of the Delaunay Triangulation.

(a) Voronoi Tessellation for the mission region. **(b)** Delaunay Triangulation of the mission region.

?⟨sub@fig:delaunay-tri⟩?

Figure 1.0.2: Voronoi Tessellation vs Delaunay Triangulation.

oronoi-delaunay⟩

- **Grids:** Grids can be thought of as special cases of Voronoi Tesselation in which the generating points are extremely regular and dense [45, 124, 8]. Regularity can be exploited for pathfinding. In Figure 1.0.3 an extreme case of regularity, "Manhattan Distances", is shown. The pathfinding in such cases is simply finding the permutation of required directions as it is shown in Figure 1.0.3.

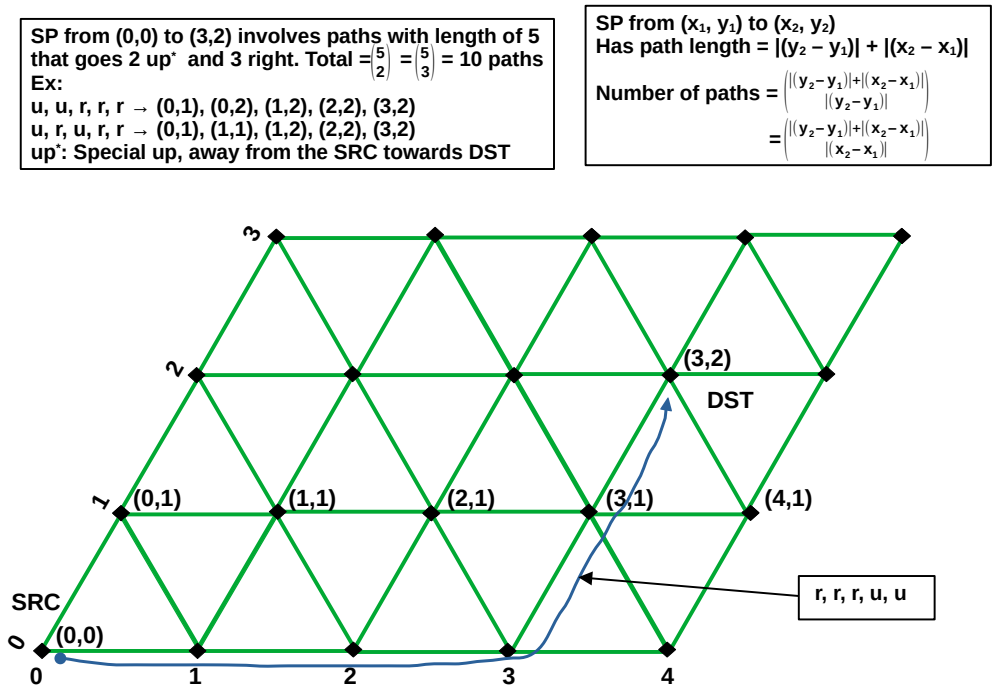


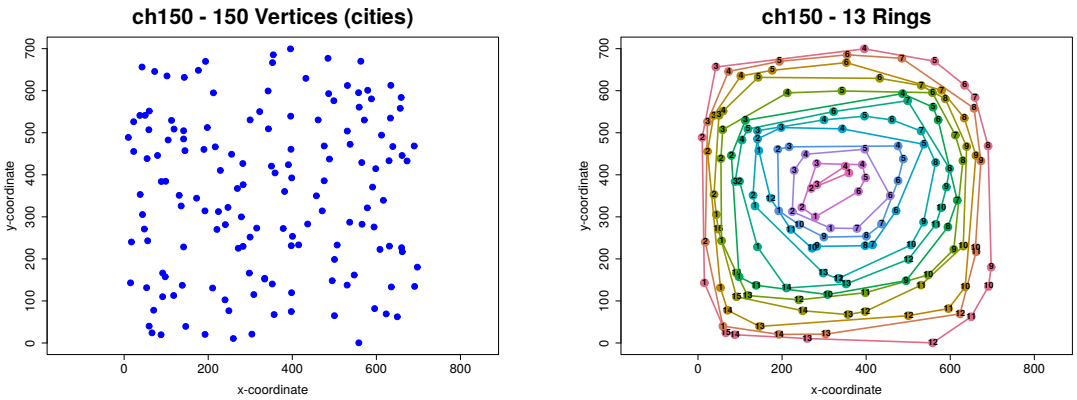
Figure 1.0.3: Simple pathfinding with Manhattan Distances on Triangular Grid.

⟨fig:manhattan⟩

- **Convex/Concave Hulls:** These geometric structures are useful for sorting vertices topologically and distance-wise [123, 12, 119]. The custom TSP algorithm, concaveTSP, that is designed for the boat rescue case study construct concentric concave hulls before merging them into a single tour. In Figure 1.0.4a example vertex layout from the TSPLIB dataset "ch150" [2] is shown. After the construction of the concentric convex hulls, the "vertex leveling" and the "clockwise sorting" can be seen in Figure 1.0.4b.

- **Circle Packing (CP):** The projection on the ground of the conical wave transmission from the drone is assumed to be circular. As the drone goes higher the radius of the projected circle on the ground increases. In the coverage optimization, the covered region portion by a single drone can be modeled as a circle associated with the altitude of the drone as it is shown in Figure 1.0.5.

[2] http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsp/index.html



(a) Vertex layout of dataset ch150 (150 city problem).

(b) Using vertices iteratively for concentric convex hulls.

?⟨sub@fig:ch150-vtx⟩? ?⟨sub@fig:ch150-rings⟩?

Figure 1.0.4: Vertex leveling and clockwise sorting after the construction of concentric convex hulls.

?⟨fig:ch150⟩?

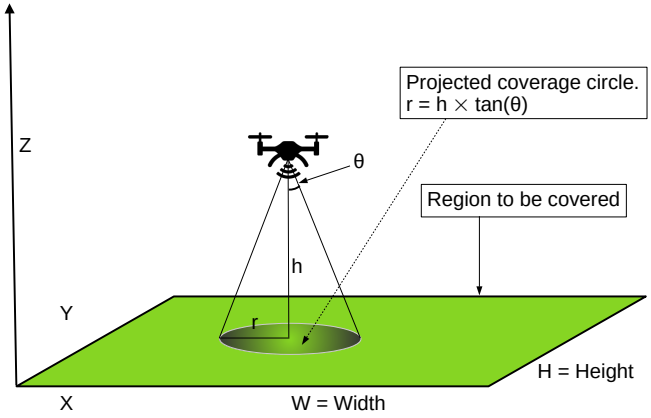


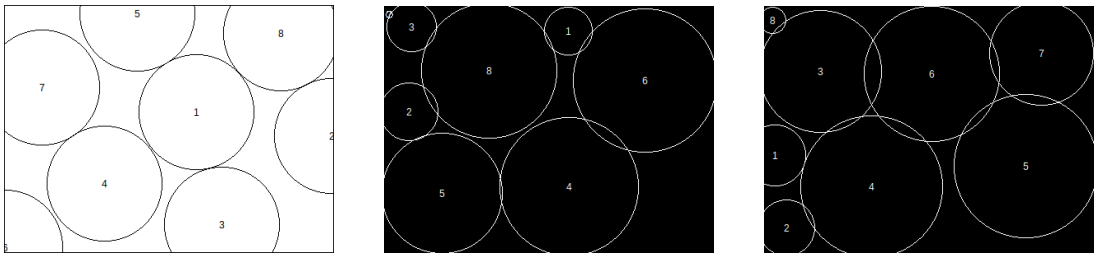
Figure 1.0.5: Drone altitude - projected circle on the ground relationship.

ingle-drone-par⟩

The CP algorithm is a fast algorithm that can be used to "pack" circles with a certain radius into a given polygon (rectangular) [28]. In the thesis, the CP algorithm is used to provide fast and "good" (min overlap possible) initial solutions for Evolutionary Algos (EAs) in region coverage optimizations. This method is a helpful heuristic that can give the EAs a good starting point for their search for the optimal solution. In Figure 1.0.6 the improvement for the GenSA algorithm is summarized.

For other advanced geometric structures and their applications on the topic of optimization, the reader is advised to look at books [114] and [65]. The works in [10] and in [3] are excellent studies summarizing how geometric configurations of the entities can be exploited for better optimizations. The thesis work in [138] stated that by looking at the geometric settings of the entities involved in the optimizations one can solve many $\mathcal{NP}$-Hard combinatorial optimization problems efficiently or in a well-approximated way.

We applied the geometric structures that are listed above to several $\mathcal{NP}$-Hard combinatorial optimizations problems in the thesis. Real-life case studies are presented and the heuristics



(a) CP for 8 drones supplied as initial solution to EAs. Coverage=75.6%

(b) GenSA result without using the initial solution. Coverage=76.96% (189 secs).

(c) GenSA result after using the initial solution. Coverage=84.85% (179 secs).

?⟨sub?⟨fig:cp-init⟩? ?⟨sub?⟨fig:gensa-isoln⟩? ?⟨sub?⟨fig:gensa-no-isoln⟩?

Figure 1.0.6: The initial solution offered from the CP method and the improved result after using it.

⟨fig:cp⟩

are designed to obtain better optimizations. We tried to exploit the geometric regularities, introduce regularities where it was possible and necessary, and look for synergies between the geometric settings and the necessary heuristic optimization algorithms. The thesis presented experimental analyses of the proposed heuristics. Several probabilistic and geometric analyses are presented. Proofs are given for several geometric heuristics.

1.1 Research Methodology

The thesis is an example of Quantitative Research that followed the Positivist paradigm. The Ontological view for the proposed optimization frameworks was that the "optimum" value exists and Epistemologically can be measured and can be known. However due to "$\mathcal{NP}$-Hard" nature of the problems the "optimum" is difficult to be found. This view guided the research work in the direction of finding heuristic approximation methods. Overall research paradigm can be summarized in the following itemized list:

- **Quantitative research**: Benchmarking, metrics, measurements.
- **Positivist research paradigm**:
  - **Ontological view**: There exist an "optimum" that can be measured and known (universally quantifiable).
  - **Epistemological view**: The "optimum" can be measured, but difficult to find $\rightarrow$ $\mathcal{NP}$-Hard problem $\rightarrow$ **Approx. algorithms**. (Von Neumann Arch & binary logic are assumed).
- **Experimental methodology**: Algorithm benchmarks.
- **Statistical verification for measurements**: Results are verified/explained with statistical tests (T-test, distribution fitting).

The activities related to the research process can be listed as:

- Literature Review
- Determination of research gaps
- Discussion of research gaps and research questions for novel contributions
- Theoretical analyses for methods and proofs
- Implementation of proposed models and methods
- Experimental verification of proposed models and methods
- Statistical verification of the results

The literature review is carried out on the related topics by using online academic databases. The following is a partial list of such online databases:

- Scopus
- IEEE Xplore
- ScienceDirect
- JSTOR

The review phase is followed by the determination of the research gaps in the related topics. The research gaps are discussed and novel methods are proposed. The proposed methods are implemented in models we designed and verified in proofs and in experimental settings. In order to communicate and share our research, we prepared articles and submitted them to academic conferences and journals. All of the material in the thesis are published in peer-reviewed proceedings and journals. The complete list of the published articles and "Code Availability" on the thesis research is given on page 123 in Appendix B. Theoretical proofs are given and Experimental Methodology is used to assess the performance of the proposed methods. Statistical tests are also utilized to verify the statistical significance of the benchmark measurements.

1.2 Research Questions, Research Gaps, and Contributions

The thesis fundamentally addressed the limited onboard energy of the EVs, more specifically drones. The core of the research work of the thesis aimed to propose novel optimization frameworks in which drones can be operated more effectively. The initial fundamental and also repeating research questions (Meta Research Questions) of the thesis were:

- **What are the basic types of drone-based operations?**
- **What are the limitations of drone-based operations?**
- **What are the "Research Gaps" in the current state of the works related to the topic?**
- **Can we propose novel methods for effective use of drones in such operations?**
- **What are the basic elements and parties involved in drone-based operation optimizations?**

These questions gave us a direction in which we can narrow the focus of research further. We selected two representative drone-based operations for research. Namely, the Coverage which involves the static operation of drones, and the Pathfinding which involves finding an optimized path to visit necessary entities in the mission region. The limited onboard energy storage (e.g., battery) of the drones is the main optimization objective for more effective operations.

The goal of the region coverage research was to cover the mission region with the available drones so that communication can be provided to any point in the region in case of natural disasters or temporary events. Drones and the region (entities in the region) can be considered as the two parties for the coverage operations. For the coverage operations topic, we have determined the following limitations from the current related work ([6, 94, 154, 21, 7]) and we listed our contributions for each:

- (In some works) **Single-party optimization, mostly focused on QoS parameters:** We proposed a multi-party multi-objective optimization scheme with EA-based approximation.
- (In some works) **"Coverage Score" with conflicting objectives was not elaborated:** We proposed a scenario-based weighted scoring (normalized scores) of objectives.

- (In no work) **Load balancing with multiple BSs was not studied:** We proposed a multi BS optimization framework in which the division of the region with Voronoi Tessellation is considered.
- (In no work) **Accelerating optimization in EAs was not considered:** We proposed a heuristic for the initial solution candidate from the CP algorithm to help EA for better optimizations.

We also needed further research questions (answered in the Chapter 2 in details) in addition to the "Meta Research Questions":

- **What is the "nature" of optimization in such operations?** The region coverage is a special case of the SCP which is a $\mathcal{NP}$-Hard problem. For this reason, the implementation of the optimization requires an approximation method. In the optimization of the region coverage, there are many parties and many objectives. Drones and the mission region are the parties. Such optimization is difficult to formulate. For this reason, EAs are utilized. For EAs generally "fitness function" is required to assess the "fitness" of the candidate solution. We designed fitness value to be the weighted sum of the objectives. We also proposed "scenario-based" weights.
- **Which EAs can be used?** We identified three candidate EAs based on the literature review. Namely GA (Genetic Algorithm) [129], GenSA (Generalized Simulated Annealing) [143], and DEoptim (Differential Evolutionary Optimization) [9, 121]. In [112] GenSA algorithm is listed as "the most capable of consistently returning a solution near the global minimum". The GenSA algorithm is a version of the Simulated Annealing [89] algorithm. GenSA is single-solution based metaheuristics whereas DEoptim and GA are population-based metaheuristics [80]. GAs are generally good for combinatorial optimization as they encode parameters by using bit strings and modify these parameters with logical operators [121]. On the other hand, DEoptim encodes population members with floating-point numbers and uses arithmetic operations for "mutations". For this reason, it is good at finding global optimum given a real-valued function with real-valued parameters. It also does not require that the supplied function be continuous or differentiable. It helps when the objective function (the function to be optimized) is stochastic, noisy, or difficult to differentiate. But can be inefficient when the objective function is "smooth". Basically, it is very similar to GA but mostly designed for continuous optimization [9]. It explores the solution space regardless of its size [67]. The following funny analogy of kangaroo for the GA and SA algorithm is given in [107, p. 30]:

  > Notice that in all [hill-climbing] methods discussed so far, the kangaroo can hope at best to find the top of a mountain close to where he starts. There's no guarantee that this mountain will be Everest, or even a very high mountain. Various methods are used to try to find the actual global optimum.
  >
  > In simulated annealing, the kangaroo is drunk and hops around randomly for a long time. However, he gradually sobers up and tends to hop up hill.
  >
  > In genetic algorithms, there are lots of kangaroos that are parachuted into the Himalayas (if the pilot didn't get lost) at random places. These kangaroos do not know that they are supposed to be looking for the top of Mt. Everest. However, every few years, you shoot the kangaroos at low altitudes and hope the ones that are left will be fruitful and multiply.

- **What are the objectives for optimization in such operations? How do they relate to each other?** For the objectives, we have identified coverage percentage, overlap

percentage, overflow percentage, and the normalized total distance of the drones from the BS. Among these objectives, there are conflicting ones. For example, as the drones go higher the coverage percentage increases but the energy demand also increases. An economic coverage of the region requires drones to be closer to the BS.

- **Can we accelerate EA optimizations?** In some cases, EA iterations can be accelerated if a "better" initial solution is given to the algorithm. For this, we proposed the use of the CP algorithm that is discussed in the previous paragraphs, for the initial configuration of the drones.
- **How can we extend the optimization framework for "bigger regions"?** In the presence of multiple BSs near the mission region, parallel and "load-balanced" multi-BS optimization can be achieved. We proposed to use the points closest to the BSs on the region edges to be utilized as generating points for the Voronoi Tessellation of the mission region. Then each sub-region or "cell" can be associated with the closes BS for the individual optimizations.

In the dynamic pathfinding research, our goal was to make each point of the mission region accessible for the mission drone. Drones, CS Grid, and boats can be considered as the parties for the rescue operations. The CS Grid should enable the drone to go, perform the operation, and return to the BS in an optimal (shortest path) way. The mission region should be "covered" with the CS Grid in an optimal (min number of CSs and no blind spot in the region) way.

For the dynamic pathfinding research, we have determined the following research gaps after reviewing the related work in [135, 99, 69, 70] and we listed our contributions for each:

- Optimal CS deployment was studied **only in the context of adjustable (mobile) CS Grid**: We proposed static optimal (min number of CSs and no blind spot) CS grid geometries adjusted to drone range for complete region coverage and a novel **coverage effectiveness** metric.
- Optimal CS deployment was studied only for coverage, **synergy with pathfinding was not considered**: We proposed **a custom TSP algorithm (concaveTSP) and optimum pathfinding (redGraySP) heuristics which are synergistic with the proposed optimal CS Grid deployment**.
- In pathfinding (Fuel Constrained, UAV Routing Problem (FCURP)) studies, either the region is **assumed to be covered** or the CSs are **assumed to be mobile**: We proposed **synergistic CS deployment and the pathfinding benefiting from the regular configuration** of the CS grid.

Further research questions (answered in the Chapter 3 in details) guided our research:

- **How can we deploy CSs to cover the region?** We considered regular (no "holes" and homogenous intervals) Tri (Triangular) and Sq (Square) grid configurations.
- **How to deploy min number of CSs without any "blind-spot"?** To cover the region without any unreachable point (no blind spot) special arrangement of the CSs is necessary. The CS coverages should overlap and the inter-CS distances should be arranged according to the range of the mission drone. For a certain distance value, the CS grid provides such "optimum" coverage with the minimum necessary CSs. If the inter-CS distance is smaller than the "optimum" value CS grid needs more than necessary CSs for the coverage.

- **Can we exploit CS Grid configuration for better pathfinding?** Assuming that CSs have almost infinite energy and Boats do not have any charging facility, we can define "red edges" as the edges that the drone can only use for "one-way travel" and "gray edges" for the "duplex travel". Any path on the red edges is conditional and depends on the adjacent gray edge. However, utilization of the red edges can provide savings.
- **Can we prove that redGraySP (dynamic edges) is better than the Shortest Path (static edges)?** We provided geometric proof based on the "triangle inequalities".
- **How often can you have such red-gray edges in Tri/Sq Grid?** We provided geometry-based probabilistic analyses to estimate the probability of having red-gray edges.
- **How often can you benefit from these red-gray edges?** We provided geometry-based probabilistic analyses to estimate the probability of using red-gray edges.
- **In practice, how is Tri vs Sq Grid comparison?** Not only theoretical analyses are provided, but also through simulations the comparisons are given for Tri vs Sq Grid.
- **Can we design a fast algorithm that considers geometric configurations of boats and finds the optimum rescue order?** Multiple boat rescue situations involve priorities for the boats. If there is no priority scheme then the "optimum rescue order" of the boats for the rescue operation should be found based on the geometric positions of the boats. For this, we proposed a fast TSP approximation heuristic we called "concaveTSP".
- **What are the objectives for optimization in such operations? How do they relate to each other?** For the CS Grid, the optimum deployment requires no blind spot coverage of the mission region with the minimum number of CSs. We also proposed a novel metric called "Coverage Effectiveness" (covered unit area per CS) of the grid. The pathfinding operation requires drones to use the minimum energy or shortest rescue tour. On the other hand, for boats, if there is no priority scheme, the average waiting time of the operation should be minimized. In the proposed framework we used a novel objective called AWD (Average Waiting Distance) which is independent of the drone speed.

1.3 Structure of the Thesis

The thesis consists of four chapters. Namely Introduction 1, Region Coverage 2, Pathfinding 3, and Conclusions 4. In the Introduction, the thesis presents general attributes of the research work. Chapter 2 presents the work on region coverage for single BS and multi BS schemes. Chapter 3 presents the research on optimized CS Grid, the proposed Pathfinding heuristics, and the proposed TSP algorithm. Chapter 4 lists contributions and impacts of the thesis.

2. Region Coverage

⟨chap:rcov⟩

"Μηδείς άγεωμέτρητος είσίτω μον τήν στέγην.",
"Let no one ignorant of geometry come under my roof."

Engraved at the door of Plato's Academy

Wireless network coverage can be disrupted by unexpected events. Temporary connectivity can be necessary for regions that are outside of the wireless communication infrastructure. One solution to that is the use of Unmanned Aerial Vehicles (UAVs) as mobile base stations. UAVs can act like temporary "range extenders" in unexpected or temporary events. Low Altitude Platforms (LAPs), like drones, are specific types of UAVs that can be used for range extension tasks. As the drones have limited energy for their operation, their deployment should be optimized to get the maximum possible coverage of the desired region with the constraints related to the capability of the overall system. Here, we proposed a novel framework for optimum coverage of the desired region with given drones by using heuristic optimization methods. Multi-objective optimization considers minimizing overlapping regions between drones, overflowing regions for the drones (coverage outside of the desired region), and flight distance of the drones from/to the base station. The trade-offs among constraints are resolved by using priority based optimization in which by setting weights the "user" can prioritize one or more constraints over the others.

2.1 Multi-Objective Priority-Based Heuristic Optimization Framework for Drone-Based Region Coverage

?⟨ijwgs2021⟩**?** The usage of Unmanned Aerial Vehicles (UAVs) as mobile base stations for urgent temporal communication and infrastructure offloading has been proposed as an efficient and economic solution in many studies. Our work focused on the temporal region coverage with UAVs. The existing communication infrastructure can be disrupted by unexpected events or it may need to be extended for temporary events like concerts or sports games. In such cases, UAVs can be deployed quickly to bring connectivity to the related region. This task requires the optimization of several conflicting objectives. As UAVs go higher they cover larger areas (requiring fewer UAVs for a given region) and at the same time, they consume more energy. The limited energy of UAVs should be considered for an optimum deployment. The proposed multi-objective priority-based optimization framework utilizes an evolutionary heuristic algorithm with a custom-designed scoring scheme to achieve such a task. The single Base Station (BS) based optimization framework is extended by considering Voronoi Tesselation of the coverage region based on the existing nearby infrastructure BSs. The Voronoi Tesselation provides "cells" in which inside, any UAV where ever be placed will always be closer to the center point of these cells, where the Virtual Base Stations (VBSs) are placed. This scheme helps UAVs

to receive the best signal from the related center point of the cell and helps load balancing of the required bandwidth for the whole mission region. In addition to that, the proposed multiple BS based region coverage scheme provides load balancing of the required data rate, especially in cases of sports or concerts where the number of clients can be demanding. Multi-objective optimization, besides maximizing the covered region, considers minimizing overlapping regions between UAVs, minimizing overflowing regions for the UAVs (coverage outside of the desired region), and flight distance of the UAVs from/to the base station. The trade-offs among constraints are adjusted by using priority-based optimization. A GUI based application is developed to customize parameters for region coverage on real-world maps. The optimization framework can be customized, by setting the weights according to the coverage scenario with this application. Depending on the requirements of the coverage scenario, one or more objectives can be prioritized or can be ignored over the others.

2.1.1 Introduction

⟨sintro2⟩

Although the wireless communication covers most parts of the Earth, there are still regions that 24/7 connection is not necessary, regions that the existing infrastructure can not be sufficient, and sometimes, regions that may be disconnected temporarily due to natural disasters. Also, temporary events, like sports or concerts that are out of the coverage of the existing infrastructure may need communication coverage out of the range of the current infrastructure. IoT applications like wildlife monitoring or forest fire surveillance are generally out of the range of the existing infrastructure. The connection for such cases is provided with low power low data rate technologies. In case a higher data rate or lower delay is needed, connecting such applications to the existing 5G infrastructure can be very costly. One of the cost-effective ways to provide the necessary coverage for such situations is the utilization of UAVs as flying BSs (Base Stations). Compared to the communications with fixed infrastructure, UAVs have salient attributes, such as flexible deployment, strong Line-of-Sight (LoS) connection links, and additional design degrees of freedom with controlled mobility. The coming of the 5G technology will be the new enabling technology for many novelties. However, this technology is not without any limitation. The coverage range is one of them. Although the 5G provides coverage of many more devices compared to the 4G, utilization of higher frequency limits the coverage range of 5G. The utilization of UAVs as flying BSs is one of the ways that can help to extend the range of 5G infrastructure. The need for an extended range for various 5G IoT applications is discussed in [5]. With the UAV based flexible infrastructure IoT deployment can be extended well beyond the range of the existing 5G infrastructure. This feature is an essential for IoT technologies involving wildlife monitoring and fire surveillance in the deep forest regions. UAV based applications for 5G cellular networks are extensively discussed in [150]. In the book, the type of communication assisted by UAVs is named “UAV assisted cellular communications”. Authors also listed the benefits of such communication schemes as traffic offloading, emergency assistance for natural disasters, information broadcasting, data collection from sensor networks. In [98] discussions on the benefits of using UAVs in the existing wireless communication infrastructure are presented. On the other hand, in [93], focusing on the future trends in using UAVs specifically for 5G communication infrastructure, UAV based infrastructure is compared with the fixed infrastructure emphasizing the advantageous attributes of UAV based communication infrastructure such as flexible deployment, strong line-of-sight connection, and controlled mobility. The paper [93], on the other hand, focused on the future trends in using UAVs specifically for 5G communication infrastructure. Authors in [93] compared UAV based infrastructure with the fixed infrastructure and emphasized the advantageous attributes of UAV based communication infrastructure such as flexible deployment, strong line-of-sight connec-

tion, and controlled mobility. A detailed forecast on present and future uses of UAVs can be found in the technical report [137]. An overview is presented on the challenges and opportunities of using UAVs in wireless communication in [149]. In [106], an emergency throughput coverage case study is presented. The study proposed the utilization of drones as Unmanned Aerial BSs (UABSs).

While the focus of our study is on short term communication coverage for a specific region through UAVs, the terms "drone" and "UAV" are used interchangeably. In our study, the drones are assumed to have a conical antenna propagation pattern with an angle θ and the projection on the ground assumed to be circular region. The altitude of a drone, h is related to the radius of its footprint, r as shown in equation 2.1.1, in which θ represents the conical angle of the radiation pattern for the drone:

$$r = h \times \tan(\theta) \tag{2.1.1}$$?{eq:hr2}?

Two or more drone coverages can "overlap". Coverage of drones can "overflow" outside of the desired mission region. These attributes are shown in Figure 2.1.1 in relation to the constraints.

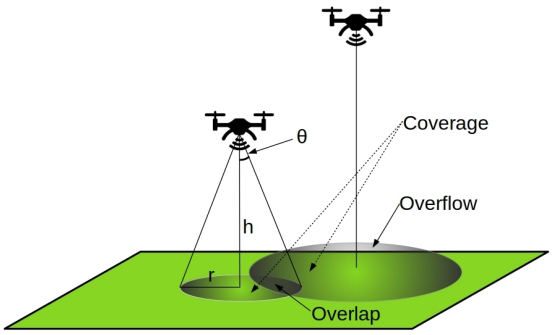
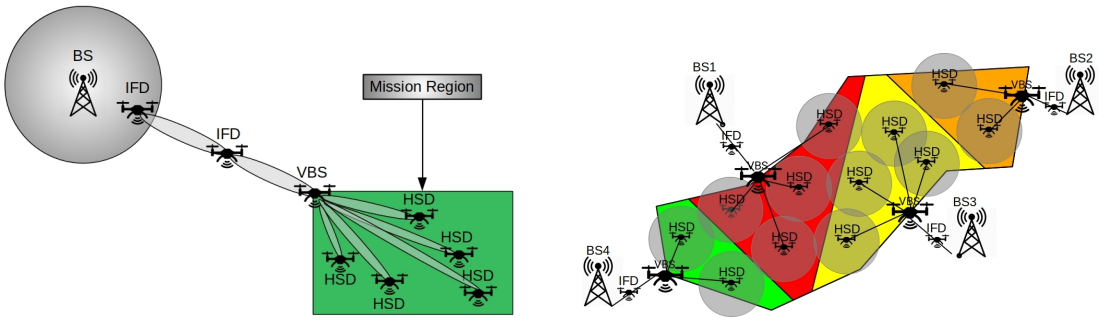


Figure 2.1.1: Individual drone parameters for coverage missions.

⟨fig:drone-config2⟩

For the proposed drone configuration, the mission region is assumed to be out of the coverage of the current communication infrastructure. In order to achieve such a coverage mission, the first thing is to bring a connection near to the region by using several UAVs in tandem. We called these types of drones Infrastructure Drones (IFDs). The number of IFDs depends on the distance between the mission region and the closest available BS. The drone closest to the mission region is named as VBS. The VBS can be positioned onto the closest point of the region bridging the drones in the mission region to the nearest BS via IFDs. The Hot Spot Drones (HSDs) are the drones in the mission region deployed for the region coverage. They are positioned onto the estimated points for optimum coverage by considering the constraints of the infrastructure. HSDs can be configured into a star topology, using the VBS as the central node. The other possibility is to arrange drones into mesh by introducing routing methods to the infrastructure. In this paper, "star" topology configuration is considered. "Inter-drone communication" is assumed to be enabled by a third radio system in addition to the drone-client, and drone control (control signals for drones) communication. Figure 2.1.2a shows the proposed configuration of the drones for covering a region that is outside of the range of the closest BS which is a part of the infrastructure. This configuration is studied in the article [87]. It is also possible to connect to multiple BSs in case of excessive demand. In Figure 2.1.2b coverage configuration with multiple homogeneous (equal capacities) BSs is shown. The coverage mission region is divided into sub-regions called "Voronoi Cells" according to Voronoi Tesselation. In this case for each BS single sub-region is assigned. A more elaborate optimization framework for region coverage is achieved in this article by integrating the multi BS

scheme with our previous optimization framework which was proposed for single BS region coverage. The standard Voronoi Tesselation algorithm assumes each "site" point to be "equal".



(a) Drone Infrastructure for covering the region with a single BS.

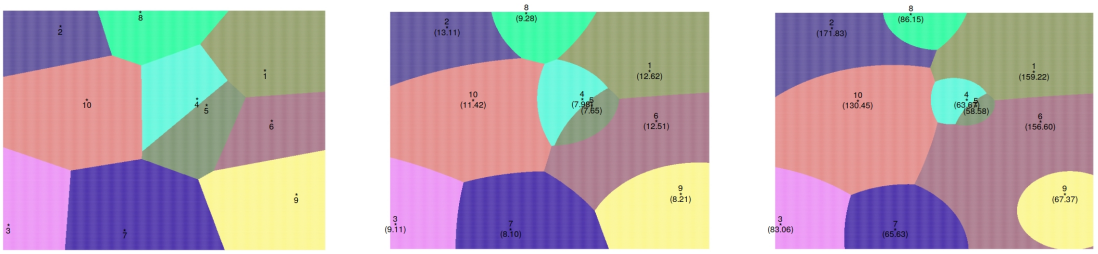
(b) Drone Infrastructure for covering the region with multiple BSs.

?⟨sub@ffigsinglebs2?? ?⟨sub@ffigmultibss-cod2??

Figure 2.1.2: Coverage configurations for single and multiple BS cases.

In the case of coverage, this means that each VBS is associated with BSs having the same properties. It is possible to associate "weights" to each site point and generate Heterogeneous Voronoi Tesselation when BSs are heterogeneous in regard to capacity. In this case the higher the capacity BS has, the greater the area it should involve in covering. In standard Voronoi Tesselation, the boundaries between the two sites are equally divided. However, if there are heterogeneous properties related to site points, the boundary points can be divided according to the proportional scheme. Figure 2.1.3a shows example Voronoi Tesselation (Euclidean distance) for 10 homogeneous site points in which the "cell" boundaries are in the midway of the lines between site points. Voronoi Tesselation (Euclidean distance) for the same configuration with heterogeneously weighted sites is shown in Figure 2.1.3b in which the boundary lines of "cells" are at distances proportional to the individual "weights" of the sites. So the sites with higher "weights" cover more area as the "cell" edges go further. One further step can be taken for considering the "square of the weights". This can be seen in Figure 2.1.3c. In this case, the points in space effected by site points similar to the "Newtonian Law of Universal Gravitation". The sites with greater weights will be more powerful. The idea here is to consider the effect of site points in linear or in higher-order depending on the nature of the function that coverage is based on. This can be useful to model BSs with different capacities. For our study, we used a homogeneous tesselation.

Voronoi Tesselation has many application areas, especially in communication technologies. Their property to divide the application region (given the site points) into sub-regions in which every point (in the associated sub-region) is closest to the associated site point is very useful for optimizing many cost functions. Example applications related to this property in the mobile communication field can be found in papers [128, 144, 145, 152]. The properties of Voronoi Tesselation are used in intra-cell handover methods in communication networks [11]. The history and applications of Voronoi Tesselation can be found in [74]. The paper [13] presents a historic survey on the Voronoi Diagrams and their various applications from the perspective of Computer Science.



(a) Voronoi Tesselation for homogeneous "site" points.

(b) Voronoi Tesselation for heterogeneous "site" points. Normalized weights (percentages) are shown for each site.

(c) Voronoi Tesselation for heterogeneous "site" points. Normalized squared weights (percentages) are shown for each site.

?⟨sub@fig:homo⟩?

?⟨sub@fig:hetero-normal⟩?

?⟨sub@fig:hetero-sq⟩?

Figure 2.1.3: Voronoi Tesselation alternatives for different coverage schemes.

The proposed framework basically tries to find the optimum locations of the drones for the maximum coverage of the mission region by considering several objectives at the same time. The objectives in our study, include maximum coverage area, minimum overlap and overflow, and minimum total flight distance of the HSDs from the VBSs. The prototype GUI application that is developed can propose the necessary number of drones for complete coverage given the average altitude of mission drones. It can also set the average altitude value given the number of drones assigned for the mission. The prototype GUI application is shown in Figure 2.1.4. For the coding various R(https://www.r-project.org/) packages are used. The GUI is developed by using "shiny"(https://shiny.rstudio.com/) framework of RStudio(https://rstudio.com/) and it can be deployed at "shinyapps"(https://www.shinyapps.io/) site.

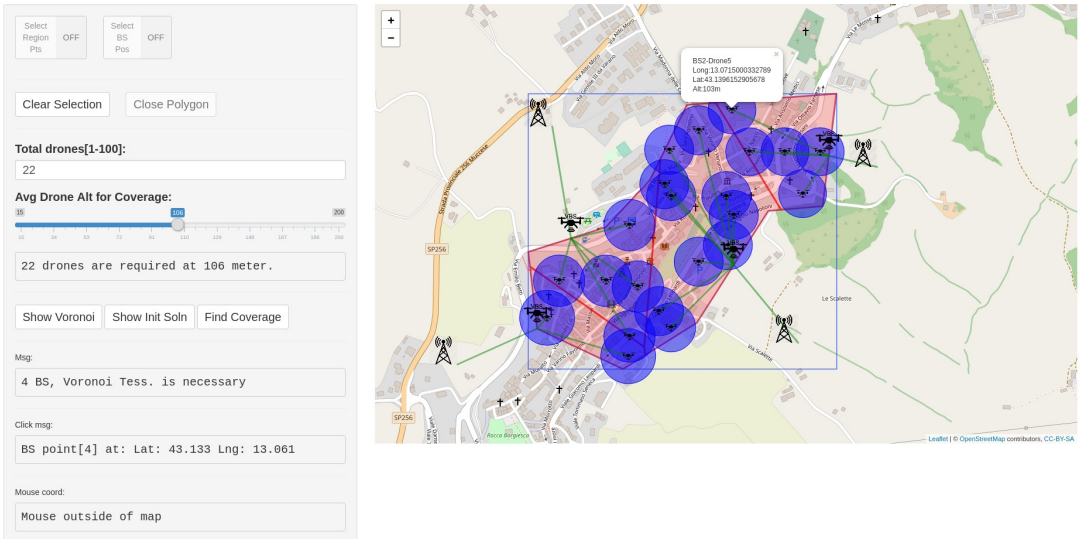


Figure 2.1.4: The prototype GUI developed for the proposed framework.

⟨fig:gui2⟩

The proposed framework takes the number of drones that can be supplied to the coverage mission as input and tries to find the maximum possible coverage of the region with desired constraints. The main novelties of our framework regarding the region coverage research are as follows:

- The utilization of the Multi-objective priority-based heuristic optimization framework. We proposed an optimization framework in which multiple objectives can be combined

for optimum region coverage.

- Voronoi Tesselation based multi-cell coverage for load-balanced multi-BS coverage. Hexagonal cells are widely used in wireless communication infrastructures for positioning BSs. This scheme is a special case of the Voronoi Tesselation in which homogeneous BSs are placed regularly. We proposed a generalized Voronoi Tesselation scheme in which not only irregular positioning of the BSs are considered, but also heterogeneous BSs are considered.

- Weighted Voronoi Tesselation for coverage with heterogeneous BSs. We extended regular Voronoi Tesselation in which homogeneous "site points" are considered. Considering the heterogeneously weighted effects of the site points allows researchers to study more general configurations of the Voronoi Tesselation applications.

- Region coverage on the real-life map for longitude-latitude based coordinates. The GUI application that is developed has the ability to be deployed on the Internet and uses OpenStreetMaps(https://www.openstreetmap.org/). With an extra communication module, the application can control real drones for coverage missions.

One novelty in the implementation of the proposed framework is the transformation of the 3D drone deployment problem to a 2D circle packing/covering problem. The projection of the drone coverage on the region is a 2D circle and the radius of the circle reflects the altitude of the drone. The value of θ relates drone coverage footprint and the altitude. With this observation, the problem becomes a circle packing/covering problem with the stated constraints.

The paper is organized as follows: A review of the issues related to the use of UAVs in wireless communication is given in Section 2.1.2. The proposed coverage framework and assumptions related to the study are presented in Section 2.1.3. Section 2.1.4 contains the summary of the experimental setup, metrics and the parameters considered for the experiments, and the presentation of the obtained results. Section 2.1.5 concludes the article by providing assessment and comments on the results along with the future research directions.

2.1.2 Related Work

⟨srelwork2⟩

In this section, the theoretical and technical aspects of coverage problems with UAVs are discussed by presenting an overview of various papers. The review is kept in order thematically and chronologically as much as it is possible to keep the presentation in a continuous manner.

The first step in such coverage problems involving optimization with multiple objectives and parameters with big dimensional space is to look at the theoretical aspect. Whether this task is achievable in polynomial complexity or not is important. If the task is not achievable with any polynomial complexity algorithm then "approximation methods" should be applied. The theoretical aspect of the time complexity for network coverage problems is studied in papers [147, 151]. According to the study in [147], such problems are in the same class of minimum set covering, which is an $\mathcal{NP}$-Hard problem. In [151], authors stated the $\mathcal{NP}$-Hardness of a similar problem in the Wireless Sensor Network (WSN) domain, namely "Optimization Scheme of N-node Coverage in Wireless Sensor Networks". The paper presented the correspondence between the famous $\mathcal{NP}$-Hard problem, the "Knapsack", and the "N-node Coverage" problem in WSNs.

Although many studies talk about coverage, they differ in the type of coverage that is considered. Two basic coverage types differ based on the objects the coverage is aimed at. Namely, "target" based and "region" based coverage types. While the target based coverage is aimed

for several targets dispersed over a specific region, the region based coverage tries to cover the whole region when the specific locations of the targets are not known. Depending on the case even hybrid schemes can be applied. In the case of emergency situations, generally "region-based" coverage is necessary. Monitoring sensors that are deployed to report periodic data in WSNs is one of the examples in which target based coverage is favored. Further discussions on the similar classification related to WSNs can be found in [18].

In the past studies, researchers focused on various aspects of the coverage problems and they presented detailed analyses on various concepts. In [6] an analytical approach for finding the optimized altitude of UAVs, more specifically Low-altitude Aerial Platforms (LAPs), for maximum coverage is presented. The authors proposed that the optimal altitude for UAVs is a function of the maximum path loss allowed and of the statistical parameters of the urban environment. In the analyses, region-based covering for a single UAV is considered. In [94], authors proposed the idea of using drones to extend network coverage to the areas where other relaying methods are not possible. The study presented an efficient algorithm to find optimal positions of the drones for maximizing data rate. In the study, the focus was on the Quality of Service (QoS) requirements aspects of transmission for a general UAV based coverage. The concept of coverage probability is studied in [21]. In the paper, the coverage probability is defined as the probability of providing signal-to-interference ratio (SIR) greater than the coding-modulation specific threshold SIR which is required for a successful reception at the receiver. The authors investigated the effects of deploying UAVs at different altitudes on the coverage. The article presented the derivation of an approximation for coverage probability. Authors concluded that the coverage probability decreases with the increasing altitudes of the UAVs. In [154], integer linear and a mixed-integer non-linear optimization models are studied for the "optimal drone placement and cost-efficient target coverage". For optimization, the study considered cost metrics like the number of drones and total energy consumption. However, in the paper, the multi-objective optimization framework is researched in an ad-hoc way. Finally, in the article, a proportional relationship is claimed between the altitude and the energy consumption of the drones. The authors focused on the "target-based coverage" in the paper.

In [130], the authors proposed "multi-tier" drone architecture for 5G/B5G cellular networks. The authors focused on QoS metrics and presented numerical performance analysis on them. The results of the study showed several benefits in using such multi-tier drone architecture over traditional terrestrial cellular networks for specific network load conditions. An extensive overview is presented in [111] about the use of UAVs in wireless networks. The paper provided technical information on UAVs in addition to the overview related to the usage of drones. Example use cases and challenges related to the UAVs are presented in the article. The study finally presented a list of open problems related to the use of UAVs in wireless communication.

The extensive survey in [55] is focused on UAVs specializing in the cellular communication topic. Considering consumer UAVs, authors surveyed issues like interference, use of UAVs as flying BSs, regulations for commercial use of the UAVs, and cyber-physical security of UAV assisted cellular communication. The topic of wireless coverage with UAVs for emergency situations like service disruption, natural disaster, and sudden user demand is studied in [132]. For such coverage missions, determining the number of UAVs and their positioning in 3D space are listed as major challenges in the article. The solution is proposed for emergency wireless coverage that involved mixed-integer linear program modeling. Based on this method, the authors presented an effective greedy approach that can scale depending on the different network sizes. Benchmarks against other related work and results are given for the proposed algorithm. The study focused on the QoS parameters of the transmission and assumed fixed grid locations for the deployment of the UAVs. In [7], authors proposed utilization of heuristic optimization methods like Genetic Algorithms (GA) and Simulated Annealing (SA) for target-

based coverage. The proposed method considered optimizing the number of drones and their location by considering data rate, latency, and throughput of the communication.

Our study filled several gaps in the optimization of UAV based region coverage research. It also proposed novel ways for previously utilized methods. The main contribution of our paper is the proposal of the generalized multi-objective optimization framework. In the previous coverage work, multiple objectives were not considered in the optimization methodically. Dividing a large mission region into multiple sub-regions was not considered in other optimum coverage studies so far. For our work, the coverage region is divided into cell-like multiple sub-regions for a "load-balanced coverage" with Voronoi Tesselation. We proposed enhancement for standard Voronoi Tesselation for heterogeneous BSs. This extension is proposed for improving the study in [87] for finding optimum region coverage for very large regions where several BSs are available nearby. In addition to that, for each sub-region, our study offers a flexible optimization framework for resolving the conflicting goals by using "weighted normalized constraints". Because of the different ranges of the constraints, it is difficult to combine several constraint metrics for "scoring" of the UAV configuration. However, our study offered a normalisation technique in which percentages of several different constraint metrics are combined by applying the weights that are customized for various scenarios. The flexibility is provided by setting the weights for individual constraint scores depending on the nature of the coverage mission. Generally, in multi-objective optimization, conflicting objectives make overall optimization difficult by creating trade-off situations. For example, in the minimum energy usage vs maximum region coverage trade-off, a minimum total distance of drones and maximum coverage constraints are conflicting with each other. As drones go higher (further than VBSs) for larger coverage areas, they also need more energy. In Section 2.1.3, different optimization scenarios and suggested weights for such scenarios are presented. To overcome such difficulty, we proposed a framework in which the "user" can prioritize energy by giving higher weight to the energy part (a total distance of drones from VBS in our study) in the optimization. Another novelty is the utilization of the hexagonal circle packing method for finding a quick initial solution for the evolutionary optimization algorithm. Evolutionary heuristic algorithms like GA and SA may spend so much time in searching for a "good starting" values. Sometimes they can get stuck in local maxima or minima. These algorithms start their evolutionary optimization process with random values. Using a "reasonable" sub-optimal configuration as initial the solution can make such algorithms to find a global "optimal" solution quicker without getting stuck in the "local maximum or minimum". In order to help the evolutionary algorithms in finding a better solution, we used the hexagonal circle packing algorithm. Since the projection of the coverage of the UAVs on the ground is circular, topics like "Circle Covering" [32, 118] and "Circle Packing" [28] can help in a similar way for supplying a "reasonable" initial solution to evolutionary algorithms. Similar ideas can be seen especially in the WSN literature. For example in [148], k-connectivity patterns of sensors are considered for region based coverage.

2.1.3 Proposed Framework

⟨spropframe2⟩ Some of the preliminary concepts related to the proposed framework are given in Section 2.1.1. In the following paragraphs, the technical aspects of the proposed framework will be discussed in detail and the overall optimization process will be presented.

The coverage framework constructed in our study considers the optimization of the region coverage with available BSs. In the study in [87] the proposed framework was for a single BS based small scale (about sports stadium-size) region coverage. Our new framework extends the previous framework for large scale (city size) region coverage. Also, the previous "pixel-based" scheme developed into a real-life map-based scheme in which longitude, latitude, and

altitude coordinates are considered in the optimization. The mission region expected to be large enough for a single BS that it can not serve all the coverage drones in the mission. The "user" selects the polygonal mission region and nearby BSs nearby which are outside of the mission region. The first step is to divide the mission region into sub-regions and assign each one of them to different BS in a one-to-one fashion. For this step, we proposed Voronoi Tesselation of the mission region. For each BS outside of the region, the closest points to the edge of the region polygon are chosen as the "site" points in the tesselation. These points represent VBS locations. The usefulness of the Voronoi Tesselation lies in the geometric partitioning of the mission region into smaller regions such that in each sub-region or sub-polygon every point will be closest to the associated site points. This means that each mission drone in the sub-regions will be closest to their associated VBSs (also BSs). Mathematically this property can be explained in formulation 2.1.2 below:

$$\begin{aligned} &V_k : k^{th} \text{ Voronoi Polygon in the tesselation} \\ &CV_k : \text{The center point of the } V_k \\ &p : \text{A point} \\ &\forall p \in V_k,\ \forall V_j,\ j \neq k,\ dist(p, CV_k) \leq dist(p, CV_j) \end{aligned} \tag{2.1.2}$$

?{eq:vor2}?

Figure 2.1.5 shows an example of Voronoi Tesselation with BSs outside of the region along with the VBSs on the region edges.

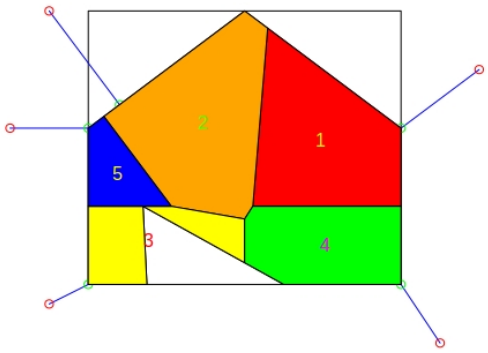
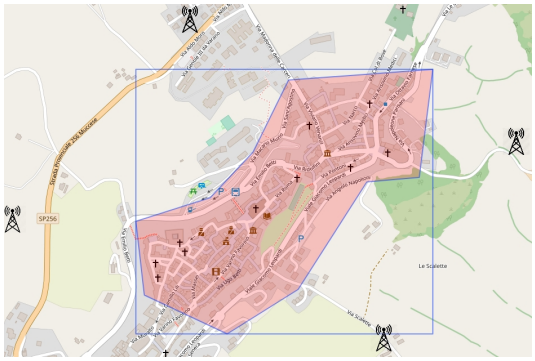


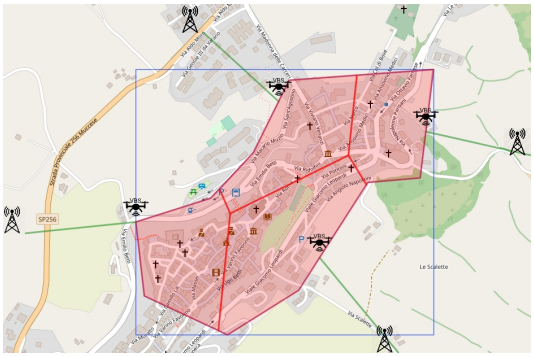
Figure 2.1.5: The region divided into five Voronoi polygons. The bounding box is drawn and BS(red circles) and VBS(green circles) locations are marked.

⟨fig:vps2⟩

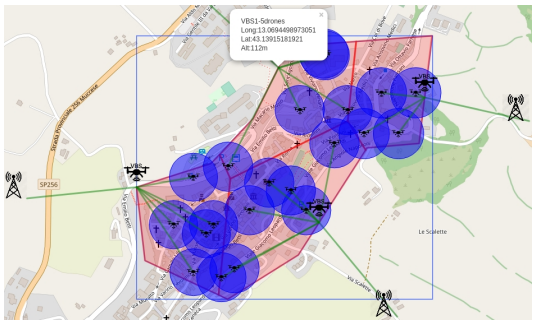
A real-life example "selected mission region" on the map of the city of Camerino (MC, Italy) is shown in Figure 2.1.6a. The Voronoi Tesselation with 4 BS is shown in Figure 2.1.6b. The initial solution supplied to heuristic optimization can be seen in Figure 2.1.6c. Resulting optimized region coverage is shown in Figure 2.1.6d. 21 drones are deployed for the mission. 100 % coverage is achieved for this scenario in which both overflow and overlap are ignored.



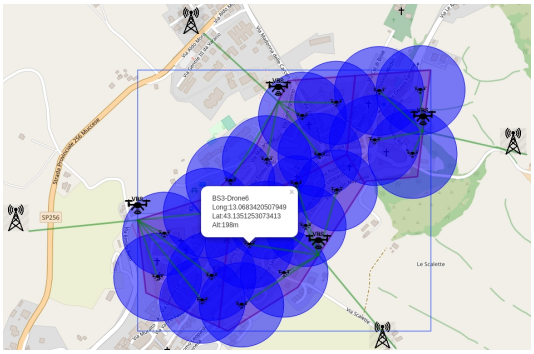
(a) The mission region and the available (or selected) 4 BSs.



(b) Voronoi Tesselation with 4 BSs and 4 VBSs.



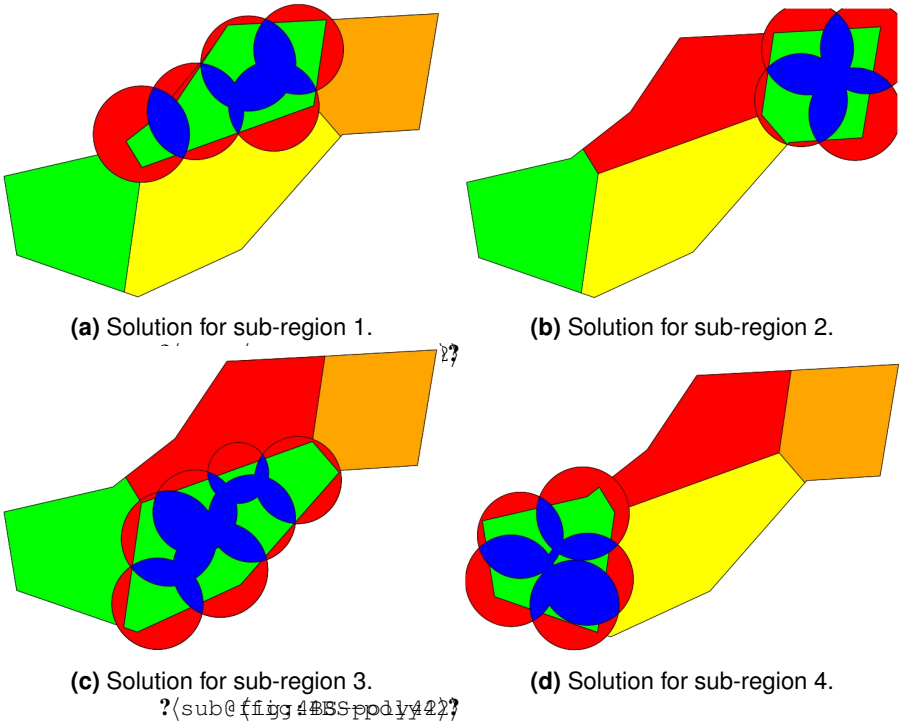
(c) Initial Solution suggested with 4 BSs and 4 VBSs.



(d) Final Solution suggested with 4 BSs and 4 VBSs.

Figure 2.1.6: Region, Voronoi Tesselation, Initial Solution, and Final Solution for region coverage with 4 BSs.

Figures 2.1.7a, 2.1.7b, 2.1.7c, and 2.1.7d each shows individual sub-region solution in a diagram in greater details.



(a) Solution for sub-region 1.

(b) Solution for sub-region 2.

(c) Solution for sub-region 3.

(d) Solution for sub-region 4.

Figure 2.1.7: Individual solution diagrams for each region. The blue regions of circles represent overlap and the red regions are for overflow.

The proposed coverage framework consists of a multi-objective optimization framework subjected to several constraints. While the main goal is maximized region coverage, with the idea of saving energy for maximizing the flight time of the drones (so the coverage time), the framework tries to find optimal positions for drones by minimizing the total flight distance of the drones. For the other objectives, the proposed framework tries to minimize overlapping coverage regions among drones and overflowing regions of individual drones. These two objectives help to maximize the coverage with the given number of drones economically. Overlaps and overflows can be regarded as minor objectives. Depending on the scenario they can be ignored or penalized by setting proper weights. There is also a threshold value set for the overlaps of individual drones. This restriction prevents the algorithm from placing one drone coverage inside or on top of another drone coverage according to the threshold value. For example threshold value of 60 means that if any drone's overlap is more than 60 percent of its coverage the solution is rejected. In Table 2.1.1, list of input parameters is given. The number of drones can be directly given to the system. Also, the user can set the "average altitude" value and let the system propose the necessary number of drones for 100 % coverage. Weight values can be positive (reward) or negative (penalty). After the user input, the initial solution is configured and supplied to the heuristic optimization algorithm automatically. The choice of the heuristic optimization algorithm is also can be regarded as one of the user inputs. The user has to know the format the algorithm accepts for the initial solution (if it has this functionality). Whether the chosen algorithm does minimization or maximisation should also be known by the user. This modification is necessary for adjusting the weight for the final score. For minimization, the score should be multiplied by -1 and for maximisation by 1.

Table 2.1.1: Input parameters for the proposed framework.

⟨tab:inputs2⟩

Parameter	Explanation
Region polygon	List of vertex (lon-lat) on the map
BS positions	List of vertex (lon-lat) on the map
Min altitude for drones	In meters
Max altitude for drones	In meters
Theta angle for drones	In degrees
Number of drones	Set by the user or the system
Overlap weight	Real number
Overflow weight	Real number
Coverage weight	Real number
Distance weight	Real number
Overflow threshold	Real number

Figure 2.1.8 shows the flowchart of the proposed multi BS region coverage optimization framework.

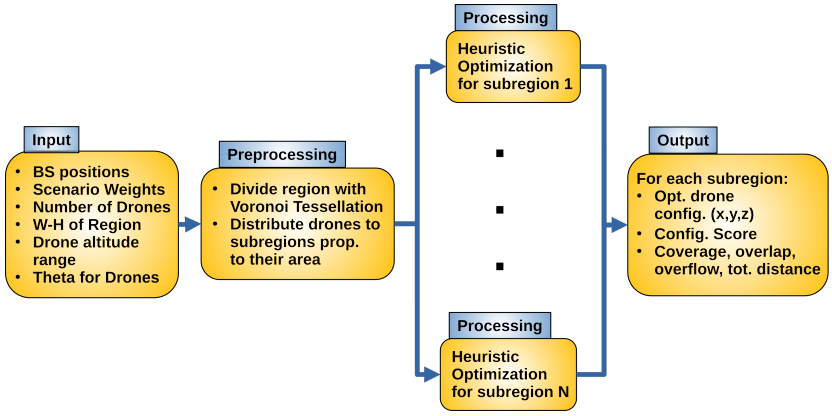


Figure 2.1.8: The flowchart of the proposed multi BS region coverage optimization framework.

⟨fig:arch2⟩

In Algorithm 1, basic steps of the whole optimization process are given. It should be noted that the system, initially proposes a "good" number of drones that is necessary to cover all the mission region by considering drones positioned at the medium altitude of their operating altitude. However, at the same time, the system lets the user change the number of drones directly or change the desired altitude (average of all drone altitudes) from the GUI. The system distributes the drones to the sub-regions according to their land area. So given the number of drones available for the mission the system offers the optimum solution with the supplied constraint weights. The system finds optimum positioning of the drones for each sub-region and finally shows the overall positioning for the whole mission region. In our study, BSs are assumed to have the same characteristics and scenario weights for each sub-region are considered to be the same.

Algorithm 1 The proposed Multi-BS "optimum region coverage" framework in pseudo code.

⟨alg:coverage-algo2⟩

Input ▷ Optimization Parameters: User supplied
Input ▷ Region Polygon: User selected on map
Input ▷ BS Positions: User selected on map of the Region Polygon
Output ◁ Drone positions and the tessellated Region Polygon
Estimation of VBS (special drones) positions:
The closest points on the region polygon edges to BS positions are chosen as VBS positions
VBSs are kept at "medium" altitude
Voronoi Tesselation:
The VBS points are chosen as "site" points for the tesselation
Weighted/Normal tesselation is carried out dividing region into sub-regions
Number of drones are distributed according to the area of each sub-regions
Initial Solution:
Drones are placed hexagonally in the sub-regions
"Extra" (out of hexagonal positions) drones are placed randomly
All drones in the same sub-region are at the same altitude
Initial solution is shown on the map
Optimum Solution:
Heuristic evolutionary optimization tries to improve the supplied initial solution
Optimization according to the "weights" is carried out for each sub-region
Optimal solution is shown on the map

Return(Optimum Drone positions and the tessellated Region Polygon)

The reader can find several different approaches related to the optimization problems in Section 2.1.2. Detailed discussions on evolutionary optimization algorithms can be found in [58]. Extensive review and detailed benchmarking on the global optimization methods (specifically in R language) for the algorithms we utilized in our framework can be found in [112]. Evolutionary heuristic optimization algorithms such as GA and SA are used in the framework we proposed. Although "annealing" algorithms do not follow exact "biological evolution" (reproduction, mutation, recombination, and selection of the best), in the thesis, because of their iterative manner they are classified as Evolutionary Algorithms. However, the reader should be aware of this difference. One good thing about these types of heuristic optimization methods is the fact that they do not require rigorous formulation and complex modeling of the problem. Crucial things that should be needed for these algorithms are the "fitness (goodness) function", the stopping condition, and the parameters with their ranges. Their function is the "intelligent search" for the optimum parameters guided by the "fitness function" supplied. The result is, generally, the approximation of the true optimum. The "user" supplied "fitness function" is used to "score" the parameter configurations. Depending on the fitness score, the mathematical framework established in the algorithm decides the "right" direction for updating the parameter configuration. The termination of this "evolutionary" process can be based on the given maximum number of iterations, on the given error margin (if the optimum value is known), on the given maximum number of iteration for "steady state" criteria (when the score does not change for n number of iterations), and on the given computing time limit. For finding the necessary maximum number of iterations, several executions are necessary. Error margin can be given if the "optimum score" for the fitness function is known beforehand. The limit for the computing time is useful when there is a need for a quick result. The maximum number of iterations for an unchanging fitness score value enveloped in a certain margin can be useful to prevent unnecessary iterations. In practice, the most important and difficult part of utilizing such heuristic optimization is the design of the fitness function. Basically, this function is supplied with the set of parameters that describe the drone configuration. As a result, the function assesses the quality of the drone configuration specified by the necessary parameters with a single value. This value or score guides the algorithm for improving the current solution in an iterative or evolutionary way. Constraints of the optimization are integrated into the fitness function design and also they are specified in input parameter ranges. For our proposed framework constraints consist of, minimum overlapping coverage regions among drones, minimum overflowing regions of the drones, and minimum total (also mean, the median can be

used) distance of the drones from the VBS. The minimization of overlaps and overflows provides further maximisation of the mission region with less number of drones. The minimum distance constraint is observed for energy-saving purposes. To get a single combined "score" out of these constraints, a form of normalisation is necessary. We applied normalisation for these constraints by converting them to percentages. Coverage is normalized according to the sub-region area. Overlap and overflows are normalized according to the covered area. For the distance, we considered the maximum possible 3D distances that drones can travel from VBSs in each sub-region and normalized the total 3D distances of the drones (drones in each sub-region) from VBSs. For different optimization scenarios, some of the constraints can be more important than the others. For example, the optimization could require maximized region coverage with no compromise. In such scenarios, the weight of the coverage percentage should be bigger. Forming a combined score from these constraint measures is achieved by normalizing (converting to percentages) and summing their "weighted" normalized contributions.

After briefly presenting the workings of the evolutionary algorithms and explaining the scoring mechanism used in the proposed framework, in the following paragraphs formulations along with explanations will be given for each constraint measure. The contributions to the fitness score come from coverage, overlap, overflow, and distance related metrics. How these metrics are extracted will be discussed below in greater detail. The formulations are for scoring the individual sub-regions. Currently, the proposed framework does not associate any global measure for the whole mission region.

Considering n drones as $d_1 \dots d_n$ with each having coverage areas represented as A_i, the intersection area of two drones d_i and d_j can be formulated as $A_i \cap A_j$ and their union can be formulated as $A_i \cup A_j$. The total area of the mission sub-region can be represented as A. The contribution of the coverage measure to the fitness score is given by the coverage percentage, which is practically the percentage of the covered area with respect to the total area of the sub-region. Theoretically, this means finding the union of individual drone coverages by considering the intersections among drones. Practically two basic methods can be followed in programming. The first one is utilizing "pixel" based labeling in which the use of matrices is necessary. The second method we can advise is the utilization of "geometric" objects and methods. According to the notations we adopted, the area of the covered sub-region can be given in the following equation 2.1.3:

$$\text{Covered Area} = \bigcup_{i=1}^{n} A_i = \bigcup_{\emptyset \neq I \subseteq \{1,\dots,n\}} (-1)^{|I|+1} \bigcap_{i \in I} A_i \qquad (2.1.3)$$?{eq:cova2}?

The formulation is similar to the N-set union formula. For calculating the "fitness score", the percentage of the covered area with respect to the total area is utilized.

The sum of all regions covered by two or more drones gives us the total overlap. The contribution of the overlap to the overall score comes from the percentage of the total overlap with respect to the total covered area. In the same way, the contribution of the overflow is calculated. In this case, total overflow is the sum of regions covered by drones outside of the mission region. These two measures guide optimization algorithm to discover "economical" coverage configuration for the drones. However, the flight distance of the drones is important for energy savings and keeping the coverage duration as long as it is necessary. For this purpose, we proposed a simple measure, total flight distance (or drone distance from VBSs) percentage, which can be calculated for energy savings. Different strategies can be applied to obtain a value that will penalize the score for longer flight distances of the drones. In our framework, we proposed the distance percentage which is a normalized distance sub-score. It represents the sum of individual drone flight differences from the maximum possible 3D flight distance in the related sub-region. The calculation of the "maximum possible flight distance" in each sub-region takes

a reference to the associated VBS position. Geometrically, it is the length of the maximum line segment that can be drawn in the polygon of the related sub-region from the VBS location. The bigger total difference means shorter drone flights so less energy consumption. In this way, the sub-score for the "distance" measure is a kind of reward. The weight for the distance percentage is generally positive for this method. The total distance percentage can also be taken as the sub-score with negative weights. In formulation 2.1.4 below the calculations are explained in equations:

$Dmax$: Max 3D flight distance in the sub-region with respect to the VBS.
$TotDist$: Total 3D flight distance of drones in the sub-region.
N : Number of drones assigned to the sub-region.
TDP : The sub-score for distance. Total Distance Percentage.

$$TDP = 100 \times \frac{(N \times Dmax - TotDist)}{(N \times Dmax)} \tag{2.1.4}$$

?{eq:maxdist2}?

Combining all the sub-scores or percentages of the measures with weights, the overall fitness score of the drone configuration can be obtained for the related sub-region. The score of the fitness function for a given drone configuration can be expressed in the following equation:

$$\text{Score} = W_c \times CP + W_l \times OlP + W_f \times OfP + W_d \times TDP \tag{2.1.5}$$

?{eq:scr2}?

In Table 2.1.2 parameters used in the equation 2.1.5 are explained. The weights are set by the "user" depending on the coverage mission type or scenario. Table 2.1.3 is given to show different types of coverage scenarios and associated weights for each. Experiments are carried out for each scenario type. Reading terrestrial static sensor values can be regarded as an example of the first type scenario. In these types of coverage scenarios maximisation of the coverage region is the most important constraint requiring low or "zero" weights for other constraints. For some coverage scenarios, the energy of the drones can not be sufficient for such missions. In such scenarios, flight distances of the drones should be considered and weights should be set for distance scores. When the number of drones is limited for the coverage mission, the overlapping and overflowing coverage regions should be weighted as penalties. This will guide the optimization algorithm in finding drone configurations with larger coverage with the deployed drones.

Table 2.1.2: Parameters for equation **??**.

⟨tab:scr-pars2⟩

Parameter	Explanation
W_c	Coverage weight
W_l	Overlap weight
W_f	Overflow weight
W_d	Total distance weight
CP	Coverage percentage
OlP	Overlap percentage
OfP	Overflow percentage
TDP	Total distance percentage

⟨tab:app-type2⟩

Table 2.1.3: Application cases and proposed set of weights.

Application type (Scenario)	W_c	W_l	W_f	W_d
S1: Max coverage, no compromise	+	0	0	0
S2: Max coverage, overlap/overflow penalty	+	-	-	0
S3: Max coverage, overlap/overflow penalty, min total distance	+	-	-	+
S4: Max coverage, min total distance	+	0	0	+

2.1.4 Performance Evaluation

⟨sperfeval2⟩

To assess the performance of the proposed framework, several evolutionary heuristic optimization algorithms are chosen to be benchmarked(64-bit openSUSE Tumbleweed Linux platform and R Version 3.6.1 are utilized on the AMD®Ryzen 9 3900X CPU based PC.) under different scenarios. Three different algorithms that represent three different paradigms are benchmarked. Related packages of R are utilized for each algorithm. Namely, traditional Genetic Algorithm (GA) [129], simulated annealing (GenSA) [143], and evolutionary algorithms like "differential evolution algorithm for global optimization" (DEoptim) [9] algorithms are chosen. One common thing for these algorithms is the evolutionary execution since they try to "improve" their initial solutions iteratively. GA is a method that imitates the natural mechanisms that are related to the "genes". On the other hand, SA is inspired by the man-made method that is used in metallurgy for decreasing the defects in metals. DEoptim algorithm can be classified as a purely mathematical method. It is an optimization method that utilizes the Differential Evolution algorithm [134, 121].

SA and DEoptim algorithms can be parallelized. However, for the GenSA algorithm, we could not see any option for paralleling the execution. The initial solution can be supplied to GA and GenSA algorithms similarly. But for the DEoptim algorithm initial solution should be supplied differently. For this reason, in benchmarks, the DEoptim algorithm is benchmarked without supplying any initial solution. These algorithms can be stopped by setting the maximum number of iteration condition. However, although the concept of iteration is the same in every algorithm, the implementations of the iterations are different. The "running time" comparison of these algorithms is not easy. But one can only get a rough idea. One problem related to this issue is the fact that the final optimum score can not be determined before the execution of the algorithm in such coverage problems. If the optimum score is not known these algorithms can not be given such score as a stopping condition. Consequently, the running time for reaching the desired state (optimum solution) can not be measured precisely. So for each algorithm reaching the desired state requires ad hoc methods. In this sense, the running time comparison should not be considered as an exact performance comparison. To overcome this problem, after some trial runs, algorithms are limited by various conditions like the maximum number of iterations, the maximum number of iterations without improvement, maximum score limit, and tolerance for reaching a maximum score.

The benchmarking is carried out in a non-GUI environment by considering single BS and single VBS. System scripts automated the benchmarks isolating the user interactions. In Table 2.1.4, a set of parameters used in benchmarks is summarized. These parameters do not reflect the real-life physical setting. But they are chosen by considering the computational constraints to provide uniform settings for comparing the methods. The VBS position is assumed to be (0,0) which is the upper-left corner of the mission region. The weights (W_c, W_l, W_f, W_d) used in benchmarks were, (1, 0, 0, 0), (1, -1, -1, 0), (1, -1, -1, 0.5), (1, 0, 0, 0.5) respectively

for scenarios 1, 2, 3, and 4 respectively.

Table 2.1.4: Parameters used in benchmarking.

⟨tab:params2⟩

Parameter	Value
Region length	400m
Region width	300m
Min altitude for drones	5m
Max altitude for drones	150m
Theta angle for drones	30°
Number of drones	2, 4, 6, 8, 10, 12

Below, Table 2.1.5 is provided to explain abbreviations and measurements used in Table 2.1.6, in Figure 2.1.9, and in Figure 2.1.10.

Table 2.1.5: Legend for Table 2.1.6, for Figure 2.1.9, and for Figure 2.1.10.

⟨tab:abbrvs2⟩

Abbreviation	Meaning
SX	Scenario number X listed in Table **??**
IS	Initial solution
DO	DEoptim algorithm
GA	Genetic algorithm
GenSA	GenSA algorithm
With/No	Algorithm with or without initial solution
Time(sec)	Running time in seconds
Cov(%)	Coverage percentage achieved for the mission region
TDist	Total flight distances of drones in meters from the associated VBS
SX # of 1^{st}	Number of 1^{st} places for Scenario number X

The results of benchmarking are summarized in Table 2.1.6. Cells with yellow background contain the initial solution statistics. Cells with green background show the best results for each category. At the bottom of the table, algorithms are ranked according to scenarios by counting the number of times they got the best score. The last row gives the overall ranking. The values for the initial solution included in the table to show if there is any improvement that algorithms (GA, GenSA) achieved. An example initial solution configuration for 8 drones is shown in Figure 2.1.9. The running time for creating the initial solution was less than a second. TDist represents the total distance of drones from the VBS. For the initial solution, drones are configured according to the hexagonal circle packing algorithm with zero or minimum overlap. In creating the initial solution configuration, overflow is ignored. From Table 2.1.6 it can be said that DEoptim was the quickest algorithm. On the other hand, GA was the best in finding more "compact" (drones closer to VBS) coverage configuration although in coverage it was not the best algorithm. The "compactness" of the drone configuration is depicted in Figure 2.1.10. In Figures 2.1.10a, 2.1.10c, and 2.1.10e on the left, the optimization process is customized for Scenario 1 and in Figures 2.1.10b, 2.1.10d, and 2.1.10f on the right, the optimization process is customized for Scenario 3. Since in Scenario 3, the distance constraint is weighted, algorithms try to bring drones closer to VBS. The other thing to note is the amount of overflow and overlap difference between the solutions offered for these two scenarios. In

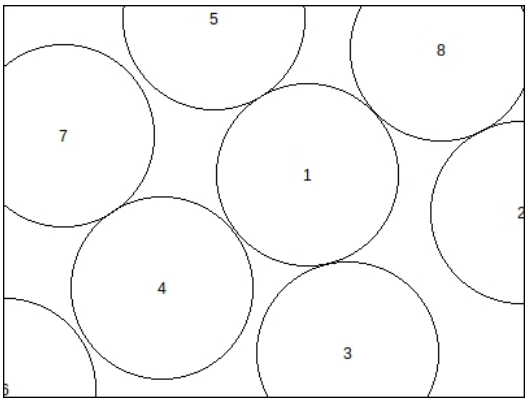


Figure 2.1.9: Initial solution for 8 drones (hexagonal circle-packing algorithm). Cov: 75.61%. TDist: 2457.35 m. Time: $<$ 1 sec.

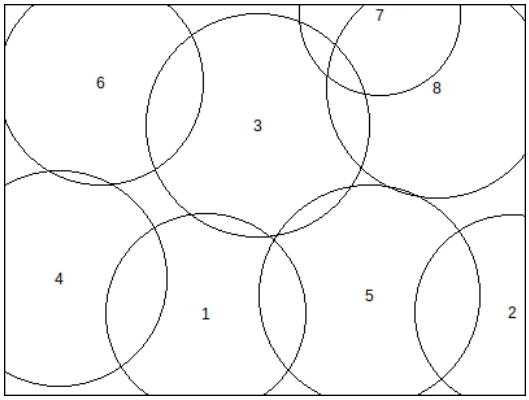
⟨fig:initsol2⟩

Figures 2.1.10b, 2.1.10d, and 2.1.10f on the right, drone coverages are mostly in the mission region and overlaps are minimized as in Scenario 3 overlaps and overflows are penalized with negative weights. Regarding the coverage, GenSA was the best in finding the highest coverage percentage. GenSA benefited from the initial solutions most among the algorithms as can be seen from Table 2.1.6. For the coverage percentage, GenSA with the initial solution showed the best performance. Whereas for the GA initial solution did not help much. But this can be said only for the specific type of initial solution configuration supplied in the benchmarks we have performed. Generalisation can not be made without trying other types of initial solution configurations.

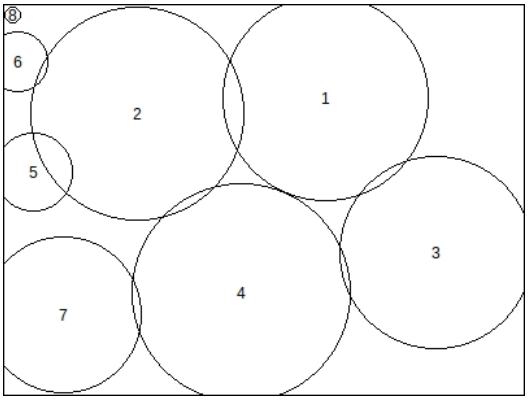
Table 2.1.6: Benchmark results for running time, coverage, and total distance of drones from VBS for each scenario.

⟨tab:results2⟩

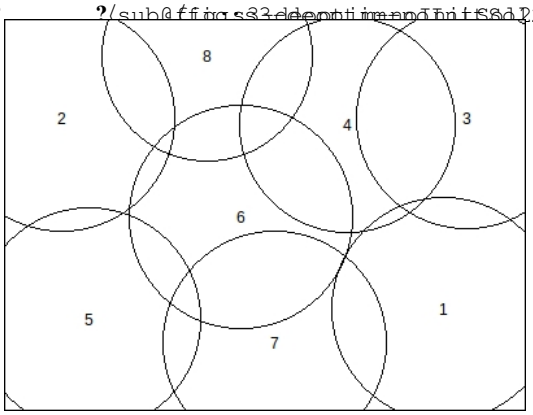
Drones	Scn	Time(sec)					Cov(%)						TDist(m)					
		DO	GA		SA		IS	DO	GA		SA		IS	DO	GA		SA	
			With	No	With	No			With	No	With	No			With	No	With	No
2	S1	13.82	209.3	361.92	743.39	729.64	21.94	40.05	40.05	40.05	40.05	40.05	475.6	565.26	564.74	596.96	666.55	597.95
	S2	15.27	196.86	207.18	712.06	719.62	21.94	40.05	40.05	40.05	40.05	40.05	475.6	652.55	608.76	627.69	636.88	540.39
	S3	39.78	163.48	178.62	716.18	699.62	21.94	40.04	37.74	40.05	40.04	40.04	475.6	508.32	500.19	509.66	507.35	508.7
	S4	30.9	202.51	226.14	711.26	710.78	21.94	39.56	39.36	39.13	39.56	39.56	475.6	493.09	488.73	484.75	493.21	493.42
4	S1	166.32	153.86	153.47	1000.78	1000.75	57.24	76.33	76.12	75.36	76.48	76.48	1322.59	1212.75	1215.67	1185.11	1232.22	1233.37
	S2	155.98	308.28	157.75	1001.01	1000.71	57.24	73.5	74.62	71.82	75.74	75.74	1322.59	1161.65	1177.85	1180.21	1201.64	1223.76
	S3	189.84	200.94	225.61	1000.92	1000.77	57.24	71.23	65.85	66.34	70.9	72.43	1322.59	1177.95	1083.5	1045.09	1097.27	1129.37
	S4	197.45	175.2	153.91	1001.12	1001.65	57.24	76.1	76.11	75.09	76.16	76.14	1322.59	1187.73	1194.66	1160.29	1185.27	1185.03
6	S1	427.51	511.38	577.94	1000.73	1000.75	69.59	95.86	95.8	95.38	96.42	96.43	1760.01	1880.97	1828.03	1851.19	1854.81	1853.23
	S2	378.26	251.65	236.42	1000.74	1000.72	69.59	87.65	83.69	86.43	87.1	89.48	1760.01	1841.02	1859.63	1753.23	1795.94	1793.11
	S3	390.51	325.48	425.98	1000.72	144.12	69.59	84.73	77.55	77.93	72.73	82.43	1760.01	1673.83	1530.65	1517.62	1275.02	1413.54
	S4	416.64	481.24	281.81	1000.76	1002.13	69.59	95.02	95.18	87.69	96.09	96.12	1760.01	1807.89	1796.11	1693.77	1813.75	1812.89
8	S1	269.4	437.34	524.35	1000.76	1000.77	75.61	94.63	98.07	99.24	99.9	99.97	2457.35	2534.07	2483.6	2431.52	2510.69	2499.34
	S2	269.91	458.04	364.65	1000.94	1000.72	75.61	78.96	83.72	83.04	90.68	89.75	2457.35	2343.94	2402.89	2270.75	2462.48	2486.55
	S3	647.31	445.35	502.83	179.23	189.01	75.61	76.98	81.8	80.6	84.85	76.96	2457.35	1688.09	2306.31	2227.58	1985.03	1638.37
	S4	694.71	372.77	394.13	1000.84	1001.92	75.61	96.14	98.37	98.08	98.7	98.32	2457.35	2120.46	2423.06	2293.7	2218.2	2178.36
10	S1	935.31	789.75	1652.59	377.51	363.53	67.29	99.98	100	99.98	100	100	3112.84	3118.6	3109.22	2981.74	3018.26	3136.28
	S2	394.58	609.55	1439.43	1000.79	1000.81	67.29	81.67	84.91	86.08	89.91	89.02	3112.84	2699.96	2796.58	2854.54	3206.29	2998.3
	S3	532	203.99	878.5	110.4	78.42	67.29	85.91	85.37	80.36	85.19	85.29	3112.84	2335.38	2561.08	2501.69	2257.09	2310.3
	S4	742.62	554.54	894.4	604.63	201.47	67.29	98.03	99.08	99.51	99.26	98.49	3112.84	2690.57	2657.05	2763.19	2520.76	2439.56
12	S1	772.97	638.99	1230.49	274.62	240.81	73.98	99.92	100	99.91	100	100	3518.39	3882.51	3617.53	3399.39	3663.7	3655.72
	S2	278.42	657.43	1016.11	1000.92	1002	73.98	69.41	86.77	84.23	89.53	89.46	3518.39	2853.03	3438	3278.39	3367.15	3230
	S3	602.03	1156.61	341.71	145.89	236	73.98	85.36	85.47	82.56	89.66	84.48	3518.39	3011.19	3423.38	2719.09	2776.8	2669.1
	S4	1296.59	744.6	528.75	174.49	192.89	73.98	97.09	99.86	98.67	97.18	98.76	3518.39	2860.62	3268.05	2947.51	2755.3	2943.88
S1 # of 1st		3	0	1	0	2	-	1	3	1	4	6	-	0	2	4	0	0
S2 # of 1st		5	0	1	0	0	-	1	1	1	5	3	-	3	0	2	0	1
S3 # of 1st		2	0	0	2	2	-	2	0	1	2	1	-	0	1	1	2	2
S4 # of 1st		1	1	2	1	1	-	1	1	1	3	2	-	1	0	3	1	1
Tot # of 1st		11	1	4	3	5	-	5	5	4	14	12	-	4	3	10	3	4



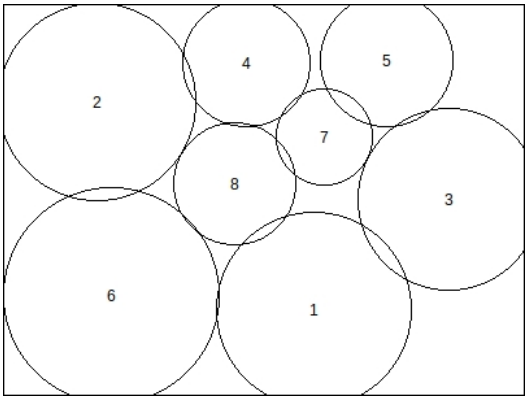
(a) DEoptim.
Cov: 94.63%. TDist: 2534.07 m. Time: 269.40 secs.



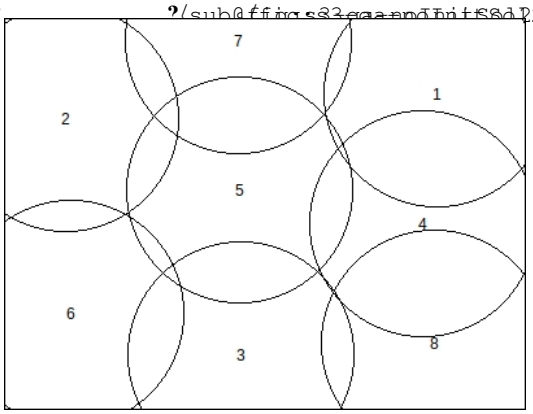
(b) DEoptim.
Cov: 76.98%. TDist: 1688.09 m. Time: 647.31 secs.



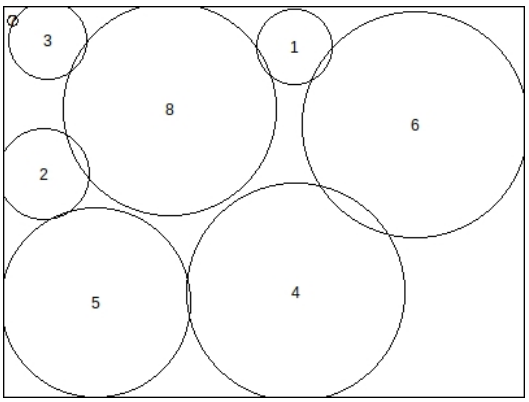
(c) GA.
Cov: 99.24%. TDist: 2431.52 m. Time: 524.35 secs.



(d) GA.
Cov: 80.6%. TDist: 2227.58 m. Time: 502.83 secs.



(e) GenSA.
Cov: 99.97%. TDist: 2499.34 m. Time: 1000.78 secs.



(f) GenSA.
Cov: 76.96%. TDist: 1638.37 m. Time: 189.01 secs.

Figure 2.1.10: Effect of weights in different scenarios.
Left: Scenario 1, weights=(1,0,0,0): Solutions for 8 drones (no initial solution).
Right: Scenario 3, weights=(1,-1,-1,0.5): Solutions for 8 drones (no initial solution).

⟨fig:s1s32⟩

2.1.5 Conclusions and Future Works

⟨sconc2⟩ The proposed optimization framework in [87] was a flexible multi-objective priority-based heuristic optimization framework for UAV based region coverage problems. Single BS and single VBS configuration were studied in that study. Evolutionary algorithms are tested for example mission regions. As a sequel to that study, the previous framework is enhanced to cover real-life coverage problems with large mission regions and with multiple BSs. Large mission regions require higher communication volume. In such situations, single VBS can not supply the necessary data rate. This problem is solved in our new framework by proposing a novel division method that utilizes Voronoi Tesselation of the mission region according to the available nearby BSs. The tesselation divides the large regions into smaller "cell-like" sub-regions in which every mission drone is closest to the associated VBS for the sub-region. Such division provides load-balanced coverage of the mission region. Not only, a novel method of Voronoi Tesselation of the large mission region is proposed for the enhanced framework, but also standard Voronoi Tesselation is generalized and extended for heterogeneous "site" points. Web deployable GUI application is developed for real-life missions on real maps. For the solution of the $\mathcal{NP}$-Hard coverage problem, our framework utilized Evolutionary Heuristic Algorithms to get an approximate solution. These algorithms do not require complex formulations but simply require parameter ranges and design of a "fitness function". The result in Section 2.1.4 compared these algorithms in various scenarios. Contrary to Pareto Optimization based frameworks in which a "compromised optimization" is searched for, we presented a flexible weight-based framework for the multi-objective optimization problems. Various application-specific scenarios are given for several coverage missions. Heuristic evolutionary algorithms, like GA and SA, are supplied initial solutions from the fast hexagonal circle-packing algorithm. In some cases, this technique helped slow algorithms like Simulated Annealing in finding higher coverages. GA did not benefit very much from the suggested initial solution. However, we propose as a future work different types of initial solutions, like drones at different altitudes. In our study, the initial solution consisted of homogeneous drone altitudes. GA was the best algorithm in finding drone configurations with minimum total drone distance. DEoptim was the fastest algorithm, although, for the implementation utilized in the experiments, the initial solution was not supplied. Another future work that can be suggested is the development of a "hybrid adaptable algorithm" since each of the algorithms benchmarked performed well in different metrics and situations. Consideration of heterogeneous BSs can be tried under the proposed "Weighted Voronoi Tesselation".

3. Pathfinding

⟨chap:pfind⟩

"Be grateful for all ordeals, they are the shortest way to the Divine."

"Words of the Mother - II", The Mother (Mirra Alfassa)

In boat rescue operations, Unmanned Aerial Vehicles (UAVs or drones) can travel long distances by utilizing the grid of floating charging stations (CSs) on the sea. To respond quickly and/or in an economic way to rescue calls, the "optimum path" from the Base Station (BS) to the boat and back to the BS should be estimated for the missions. Generally, the optimum path is the shortest path involving hops from the BS via CSs to the boat and back to the BS. However, multiple objectives can be considered for two parties, drones and boats, like priority of boats, the number of chargings for the UAV, and the average waiting time for the boats. The "drone range", which is the maximum flight range that a UAV can fly with the battery, is the fundamental parameter for the design of the rescue infrastructure. The choices for the geometry of the CS grid plays important role in the effectiveness of the heuristic we proposed. We presented mathematical analyses of the effectiveness of two different deployment strategies for the CSs based on the degree of benefit from the heuristic we proposed. Namely the triangular and square CS grids. In addition to the optimum CS Grid infrastructure, we proposed a synergistic and heuristic pathfinding algorithm called "redGraySP" which provides savings for the flight distance of the drone.

3.1 Heuristic Drone Pathfinding over Optimized Charging Station Grid

?⟨rescue2021⟩**?**

We proposed a novel optimization framework for drone-based operations which consists of the optimized 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 optimized for the drone range. We presented the case study of a boat rescue operation that is carried out in the sea. The minimization of the "flight distance" and "number of chargings" are the objectives for the drone party and the minimization 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-gray path" heuristic. A novel and fast TSP heuristic was also proposed as part of the pathfinding heuristics and its performance was assessed.

3.1.1 Introduction

⟨sintro⟩
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 [131] 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 optimized grid configurations. Such a CS grid gives an optimized (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 optimized 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 optimized 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 3.1.1a.

(a) DJI Inspire 2 drone

?⟨sub@

(b) The charging pad

?⟨sub@fig:pad⟩?

Figure 3.1.1: The consumer grade drone DJI Inspire 2 and the charging station.

?⟨fig:drone⟩?

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 [46]. 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 3.1.1b. 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 "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 [84]. 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-gray 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 [79] and VRP [83] 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 "redGraySP" heuristic (hereafter "redGraySP", red-Gray Shortest Path). We presented a mathematical analysis for different CS deployment strategies in connection with the proposed "redGraySP" heuristic method. The proposed novel TSP heuristic algorithm is analysed and benchmarked against the other standard approximation algorithms by considering various standard TSPLIB datasets and custom datasets. The savings from the proposed "redGraySP" 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 "redGraySP" 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, gray edges when there is a rescue call. After the boat is rescued these red and gray 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 optimization 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 redGraySP heuristic is assessed by estimating the savings over the base case (without heuristic). The redGraySP 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 3.1.2 we presented related work. Section 3.1.3 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 3.1.3), the proposed concaveTSP heuristic (Section 3.1.3), the redGraySP heuristic (Section 3.1.3). The experimental results are discussed in Section 3.1.3. The conclusions and ideas for future work are listed in Section 3.1.4.

3.1.2 Related Work

⟨srelwork⟩

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 [101, 133, 78]. A list of advantages of using drones in such operations is presented in [105, 109, 27]. The thesis [59] presents online and offline mission planning models for emergency situations based on the spatial interpolation methods. The paper [66] 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 [90] 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 [56] 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 [22]. In [146], 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 [117]. While the drone can charge its battery with the contact based technologies ([22]), An example battery swapping technology is presented in [108]. 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 [69, 70]. These studies proposed adjustable CS deployment strategies. As it is stated in [69], 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 [83], 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 [135]. The study [99] 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 [113, 103, 82, 104, 126, 141] 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 [97, 95, 61].

The classical TSP literature is very rich since the problem has been around for a long time. The works [52] and [29] are two seminal works from which many of the heuristic methods were germinated. The studies [75] and [127] 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 [57, 47]. 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 [39, 73], 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 [110, 60]. 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 optimized 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.

3.1.3 Proposed Framework

⟨spropframe⟩ We presented a brief overview in the Introduction 3.1.1 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 that we designed for optimized pathfinding will be discussed as well. The basic elements of the proposed framework consist of these:

- **Optimized CS grid** by using min number of CSs to cover the mission region without "blind spots". Presented in Section 3.1.3.
- The TSP heuristic, **concaveTSP**, for finding the optimum permutation to visit entities (rescue boats). Presented in Section 3.1.3.
- The shortest path heuristic **redGraySP**, for finding the "best" path from one entity (boat) to another. Presented in Section 3.1.3.

Simulations and benchmarks are designed to assess the performance of the proposed framework. Design principles and experimental results are presented and discussed in Section (3.1.3).

Figure 3.1.2 shows the representative configuration of the envisioned case study which we called the drone-assisted boat rescue operation.

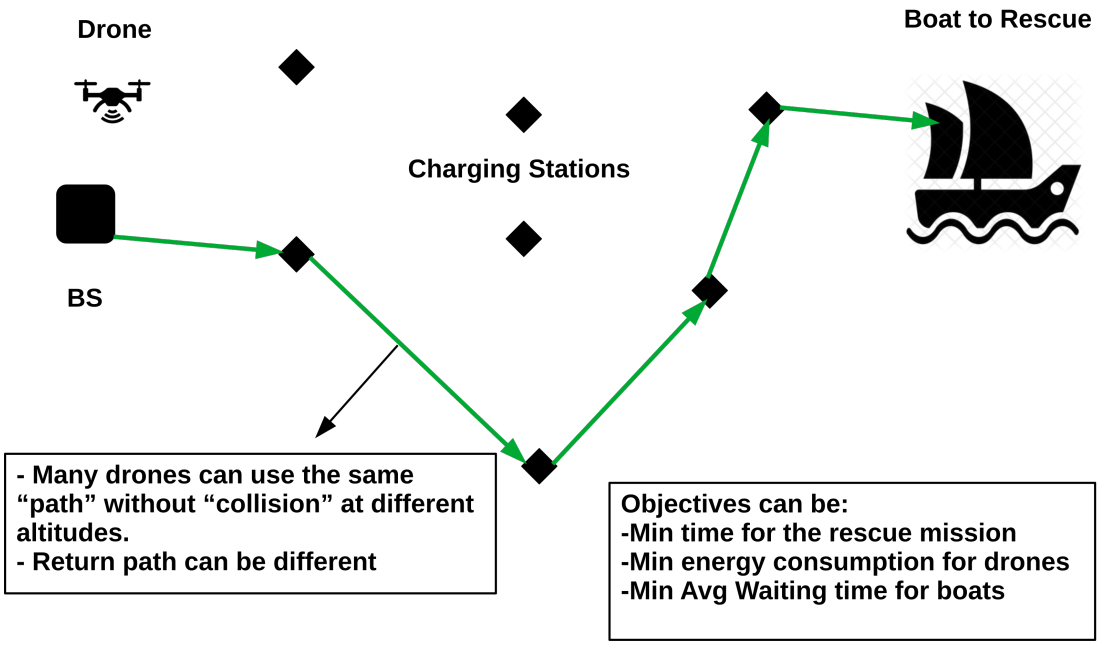


Figure 3.1.2: Boat rescue operation with drones.

⟨fig:drone-rescue⟩

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 "redGraySP" 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 "redGraySP" heuristic searches for a path-length (energy) saving pair of "red" and "gray" edges on the frog-leaping path of the drone from one CS to another on its way to rescue the "boat". The "gray" 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-gray path" is the pair of red and gray edges in which the total distance is less than or equal to the max drone range. Sometimes the drone can find a 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 "redGraySP" heuristic. While the proposed "redGraySP" 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

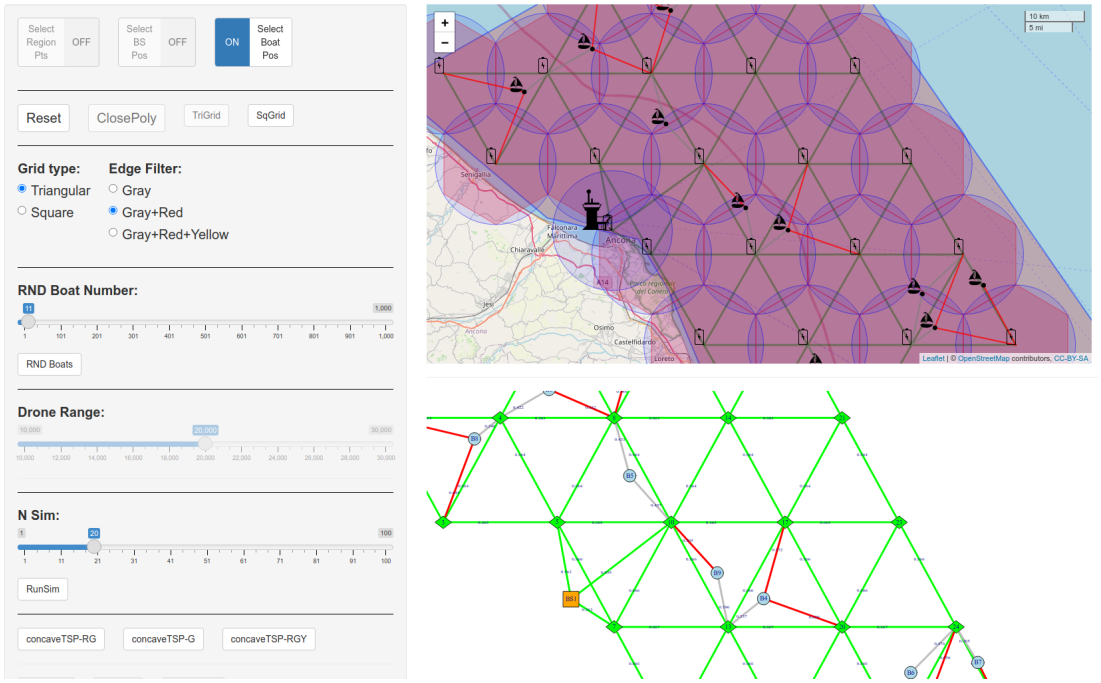


Figure 3.1.3: Prototype UI developed for the rescue system.

⟨fig:ui⟩

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 3.1.3 for this research software. The flowchart of the proposed rescue framework is shown in Figure 3.1.4. In Algorithm 2 a

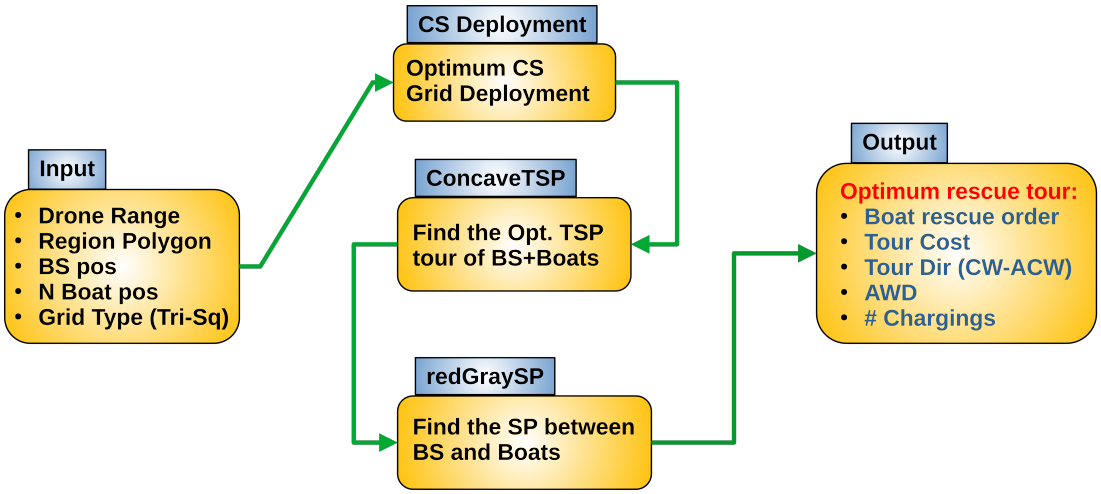


Figure 3.1.4: The flowchart of the proposed rescue framework.

oat-rescue-algo⟩

pseudo-code is given for the proposed rescue framework.

Algorithm 2 The proposed rescue heuristic optimization framework in pseudo code.

⟨alg:rescue⟩

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: $(V_1 \dots V_K)$

```
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 ← redGraySP(concaveTSPTour, Grid)
Return(RescueTour)
```

Firstly, the best TSP tour is searched with the proposed "concaveTSP" heuristic (3) in the algorithm. For this part, we utilized geometrical distance between vertices (BS + Boats) not the red-gray path distances. In the domain of the red-gray heuristic the edge/path distances are dynamical and direction sensitive. The edges of the red-gray 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-gray 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-gray shortest path heuristic, called "redGraySP" (4), the "best" rescue tour is returned. In general, the rescue tour is a multi-objective optimized tour. Namely, the flight distance, AWD, and the number of chargings are the objectives that can be weighted for optimization. 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-gray edge scheme, CW and ACW tours may differ in total rescue distance. In Algorithm 2, 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).

The CS Deployment

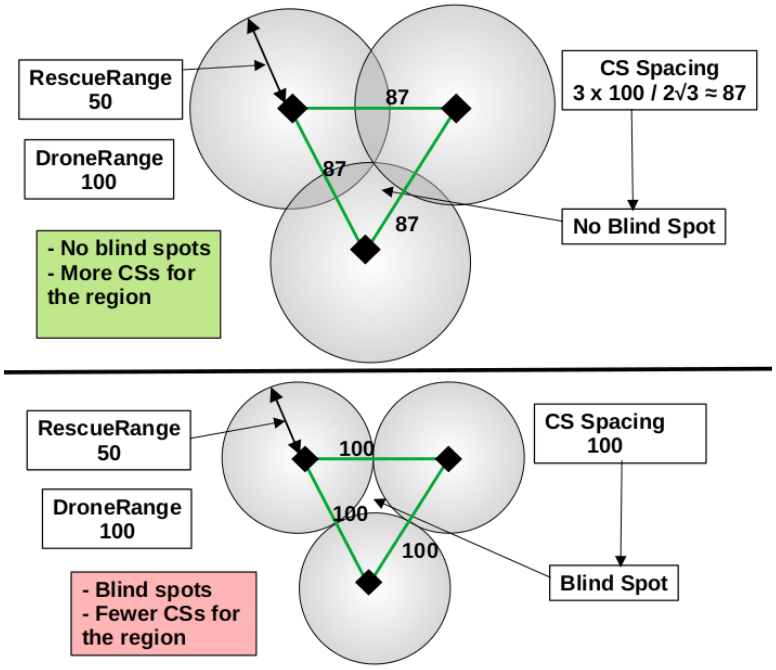
⟨sscsdeployment⟩

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

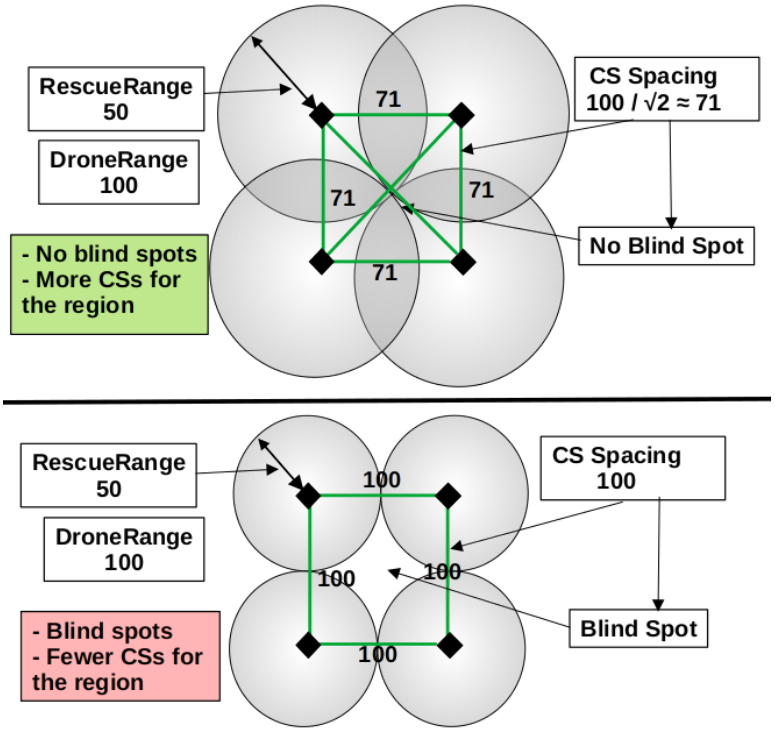
will be blind spots in the mission region. In Figures 3.1.5a and 3.1.5b, these situations are depicted.

In Figure 3.1.6 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. 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-gray path" heuristic is one of them. In Section 3.1.3 we explained in detail how the "red-gray 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-gray path" heuristic associated with the selected grid configuration.



(a) Triangular grid CS deployment properties.

?⟨sub@fig:tri-grid⟩?



(b) Square grid CS deployment properties.

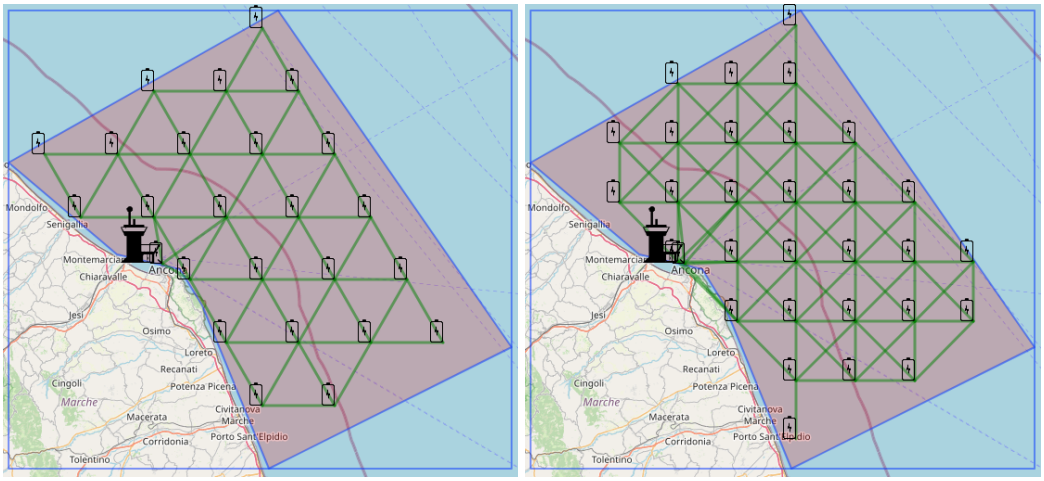
?⟨sub@fig:sq-grid⟩?

Figure 3.1.5: CS grid deployment properties.

?⟨fig:csgrid⟩?

The proposed TSP heuristic

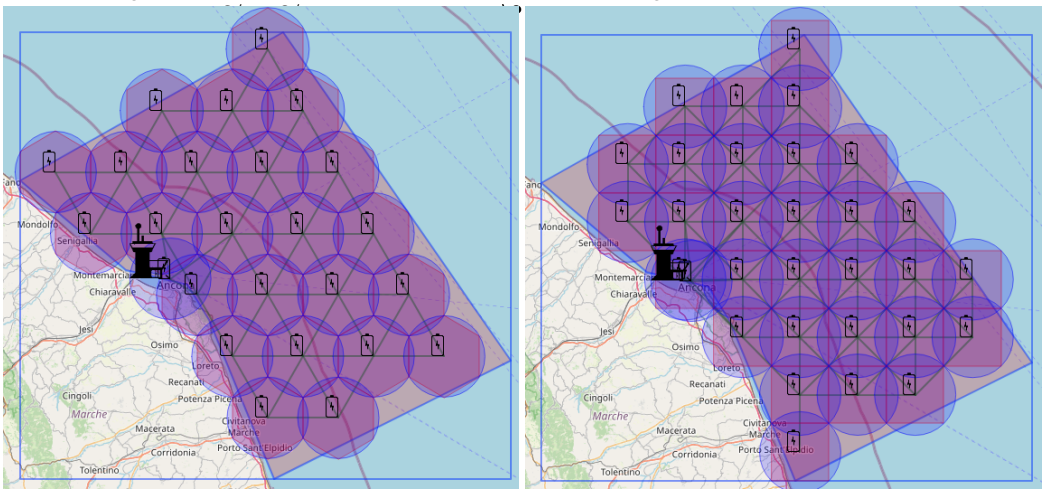
⟨sstspheur⟩ The crucial step in the rescue operation is to find the "best" order to visit the boats. Considering that the boats have equal priority, the TSP tour of the boats is the shortest tour starting



(a) Actual triangular CS Grid deployment (24 CS). AVG SP length from BS = 2.06 units.

(b) Actual square CS Grid deployment (30 CS). AVG SP length from BS = 2.07 units.

?⟨fig:ui-tri-grid⟩?



(c) Triangular CS Grid with coverage.

(d) Square CS Grid with coverage.

ri-grid-coverage⟩?

?⟨sub?⟨fig:ui-sq-grid-coverage⟩?

Figure 3.1.6: 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. unit = Drone range

⟨fig:ui-csgrid⟩

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 [119]. 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" [52]. 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 Figure 3.1.7 the flowchart of the proposed TSP heuristic algorithm is shown. In Algorithm 3 the pseudo-code is given for the proposed TSP heuristic.

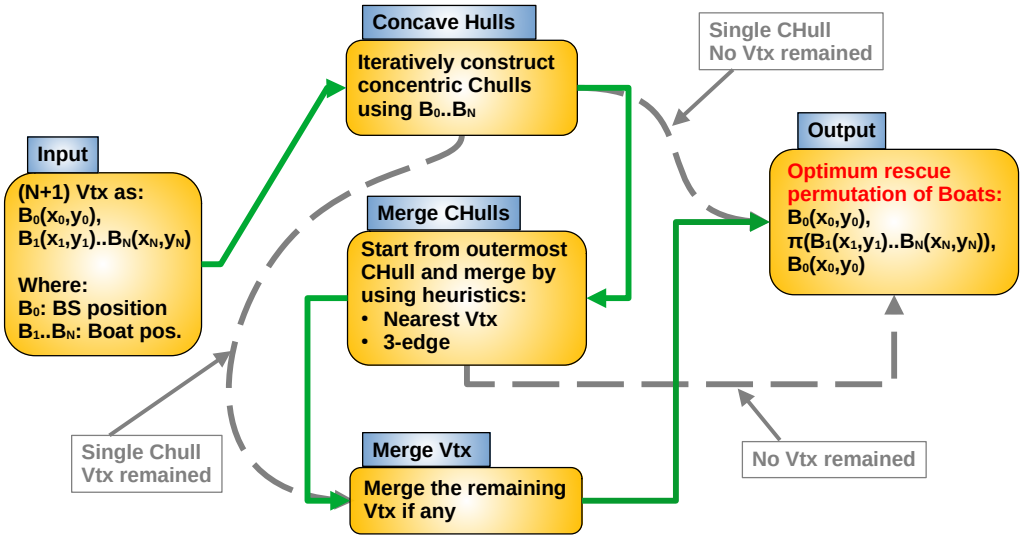


Figure 3.1.7: The flowchart of the proposed TSP heuristic algorithm (concaveTSP).

⟨fig:concaveTSP⟩

Algorithm 3 The proposed TSP heuristic algorithm (concaveTSP) in pseudo code.

⟨alg:concaveTSP⟩

Input ▷ VtxList: Euclidean vertex coordinates with numbers, $(i, X_i, Y_i), i = 1 \dots N$
Output ◁ concaveTSPTour: The approximate TSP tour, $(V_1 \dots V_N)$
Concave hull construction phase

```
1:  CHList ← NULL
2:  VtxNotVisited ← VtxList
3:  while (|VtxNotVisited| ≥ 2 ) do
4:      CH ← concave hull(VtxNotVisited)
5:      CHList ← CHList ∪ CH
6:      VtxNotVisited ← VtxNotVisited \ CH
7:  end while
8:  RemainedVtx ← VtxNotVisited
```

Merging phase: Select the nearest concaveTSPTour vertex and use 3-edge heuristic

```
9:  NSubTour ← |CHList|
10: concaveTSPTour ← CHList[1]
11: if (NSubTour ≥ 2) then
12:     for each CH ∈ CHList[2..NSubTour] do
13:         NVtx ← |CH|
14:         for each Vtx ∈ CH[1..NVtx] do
15:             concaveTSPTour ← Merge(concaveTSPTour, Vtx)
16:         end for
17:     end for
18: end if
```

Merging RemainedVtx if any

```
19: if (|RemainedVtx| ≥ 2) then
20:     concaveTSPTour ← Merge(concaveTSPTour, RemainedVtx)
21: end if

22: 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 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 3.2.3.

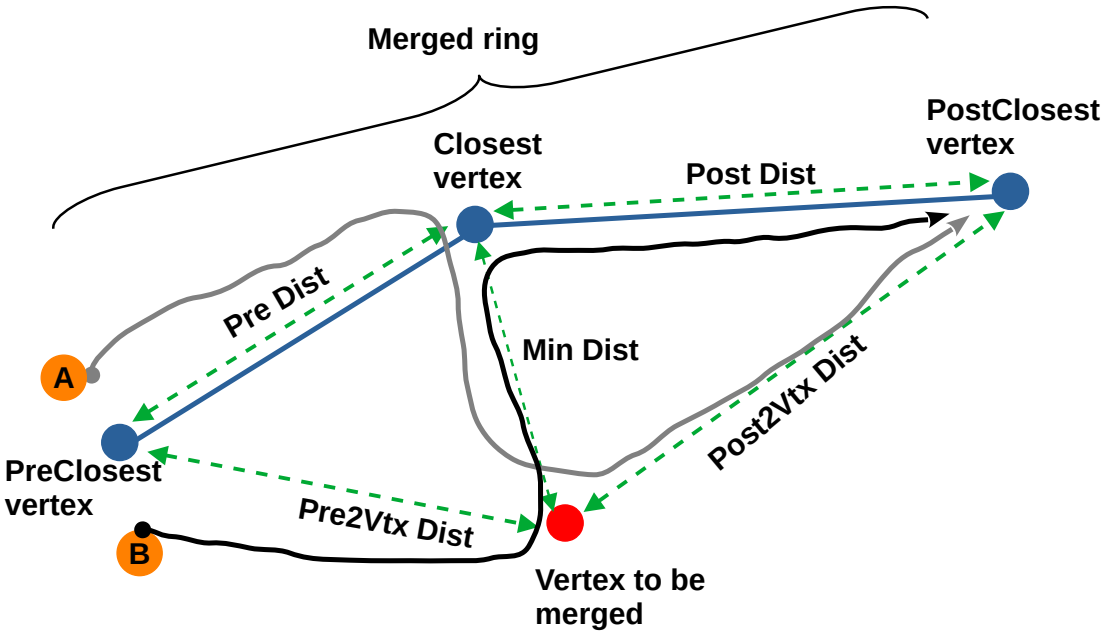


Figure 3.1.8: The 3-edge heuristic used for merging a new vertex to the ring (sub-tour) merged so far.

ge-merge-heuristic⟩

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 [119], 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 optimized 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 (3.1.3). 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 3.2.5 shows the standard TSPLIB dataset "pr76" (http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsp/pr76.tsp) with 76 vertices labelled with red colour.

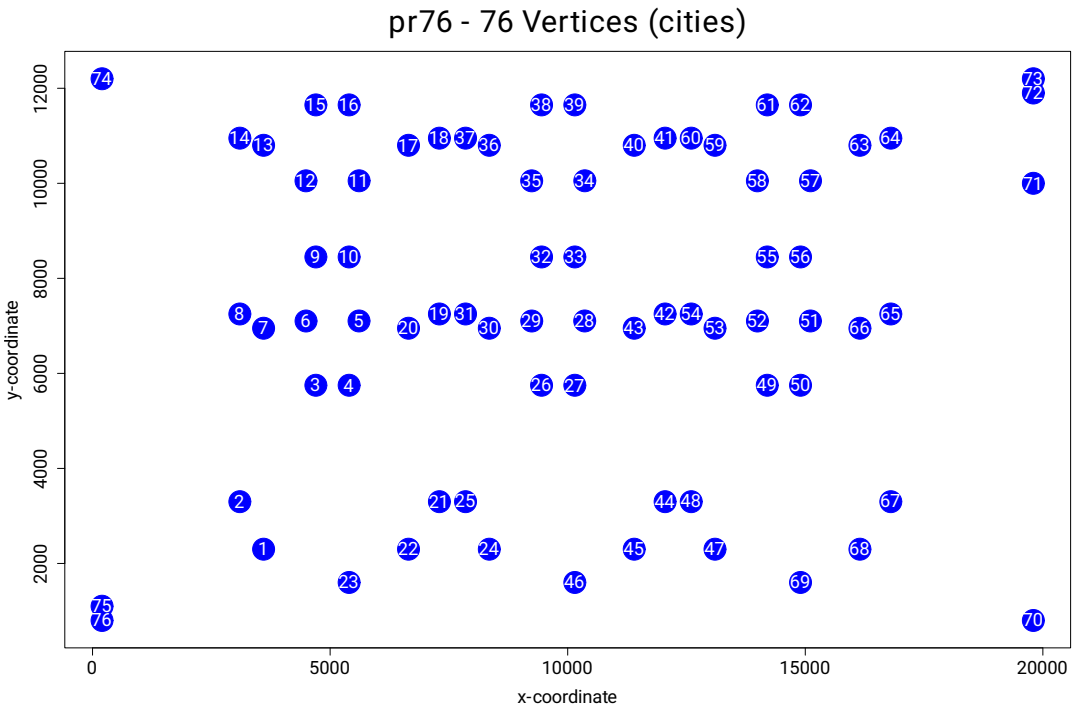


Figure 3.1.9: "pr76" TSPLIB dataset.

⟨fig:pr76-vtx⟩

Figure 3.2.7 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.

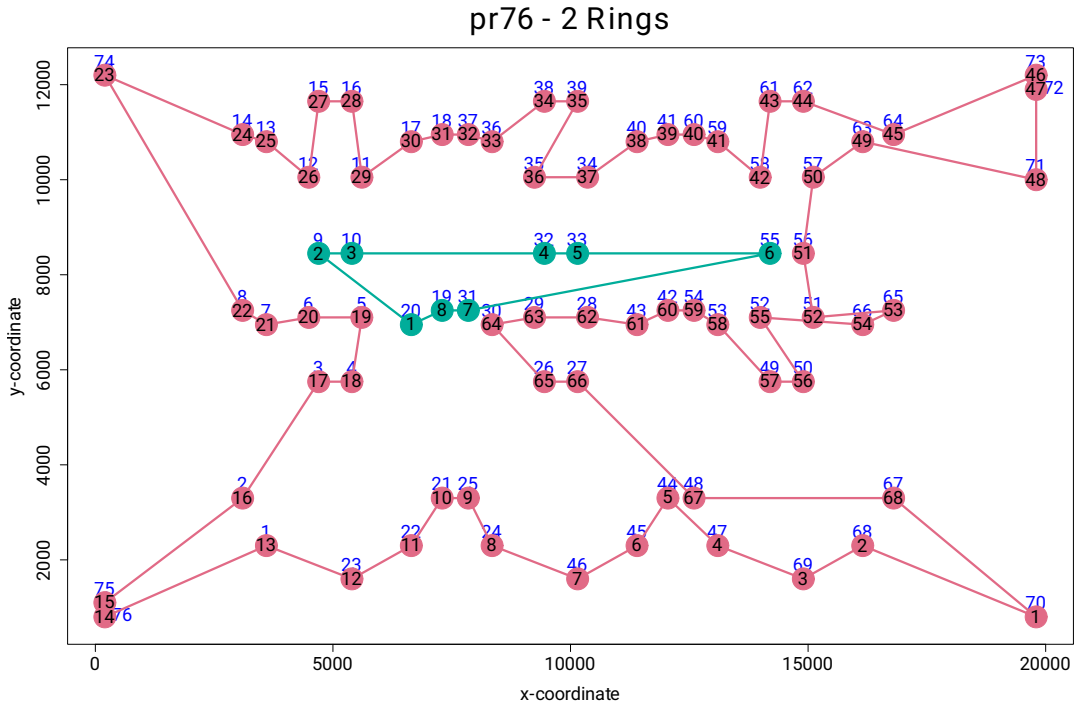


Figure 3.1.10: Rings generated by concave hull for pr76.

⟨fig:pr76-rings⟩

Figure 3.2.8 and 3.2.9 show the process of merging the first and the last point from the second ring respectively.

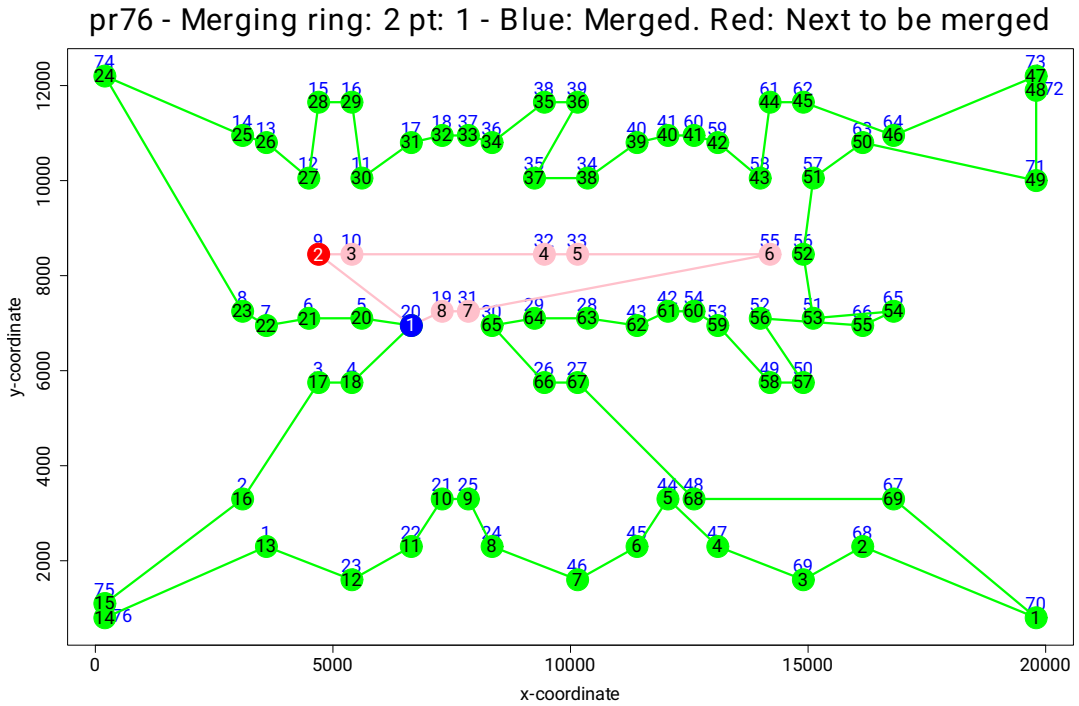


Figure 3.1.11: Merging the first point from the second ring. The next ring marked with red colour.

⟨fig:pr76-p1⟩

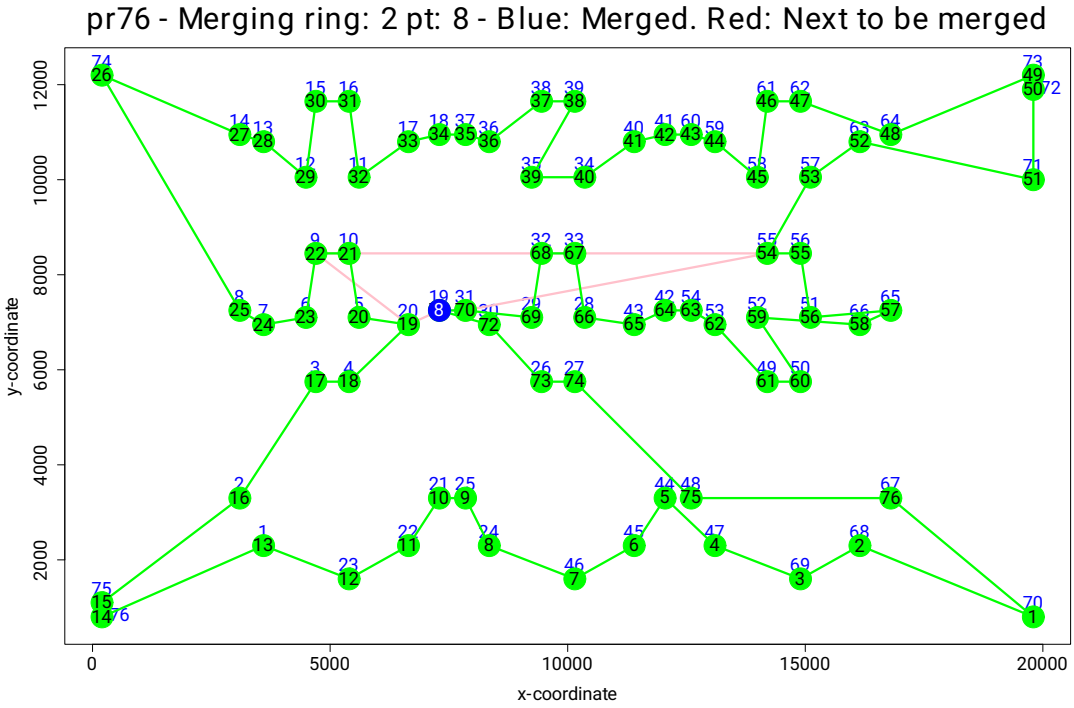


Figure 3.1.12: Merging the last point from the second ring.

⟨fig:pr76-p8⟩

In Figure 3.2.8 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 vertex number 19 (the last vertex of the second ring) is merged to vertex number 31 which was previously merged to ring 1 (Figure 3.2.9).

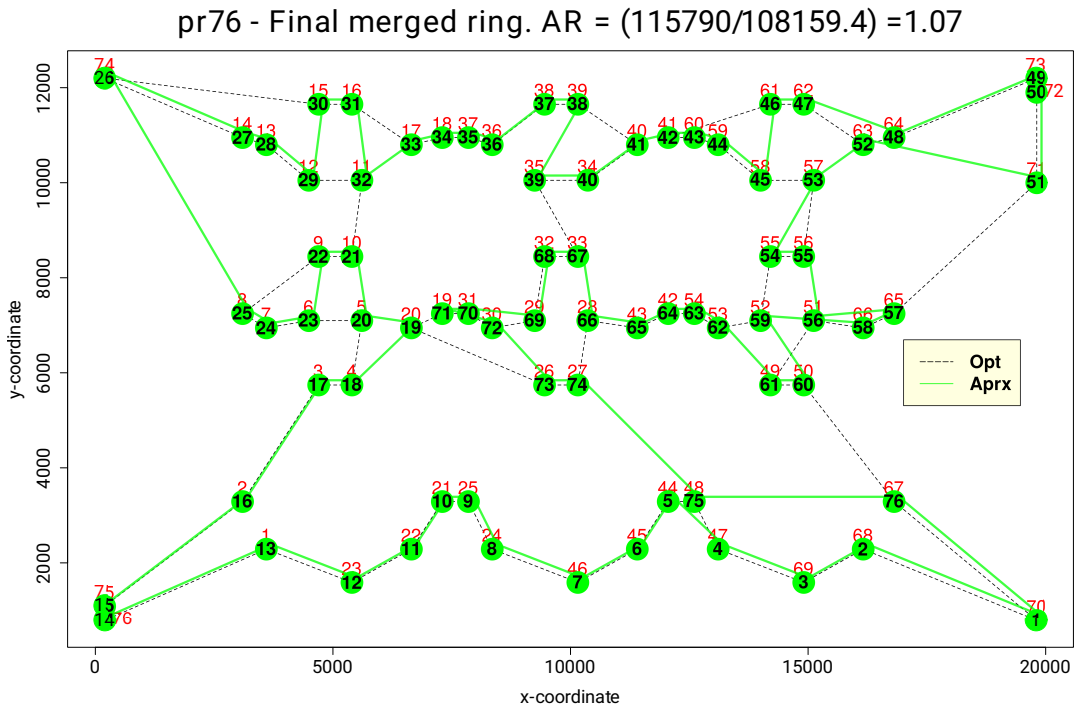


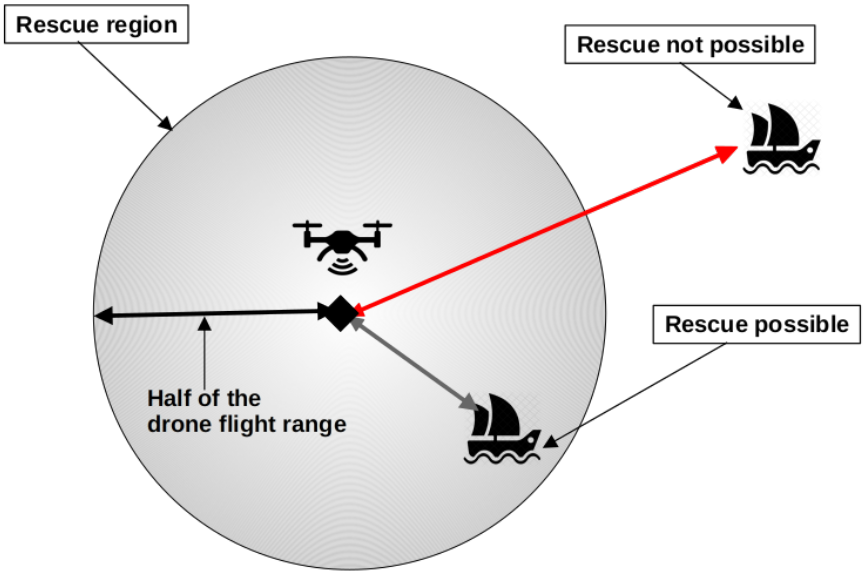
Figure 3.1.13: The final merged ring and the optimum tour for pr76.

⟨fig:pr76-final⟩

Figure 3.2.10 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.

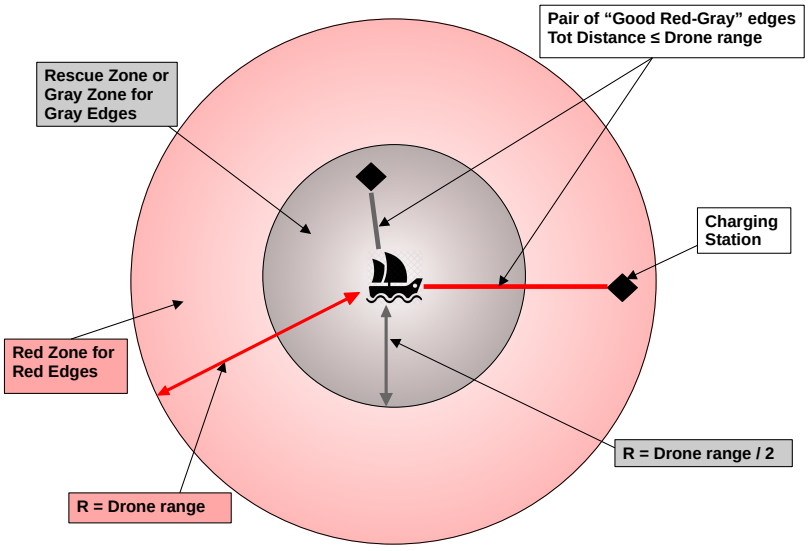
The proposed Red-Gray Path heuristic

⟨ssrgheur⟩ Assuming that the drone does not do charging on the boats, after the rescue operation it should have enough energy to 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 "gray 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 gray 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-gray" 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-gray path". These conditions related to drone range are explained in Figure 3.1.14.



(a) Conditions for rescuing a boat.

ub⟨fig:rescue-capacity⟩?



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

ub⟨fig:red-gray-ranges⟩?

Figure 3.1.14: Rescue conditions and red-gray edges.

fig:rescue-cond⟩

In Figure 3.1.15 the situation of "good red-gray path" is depicted.

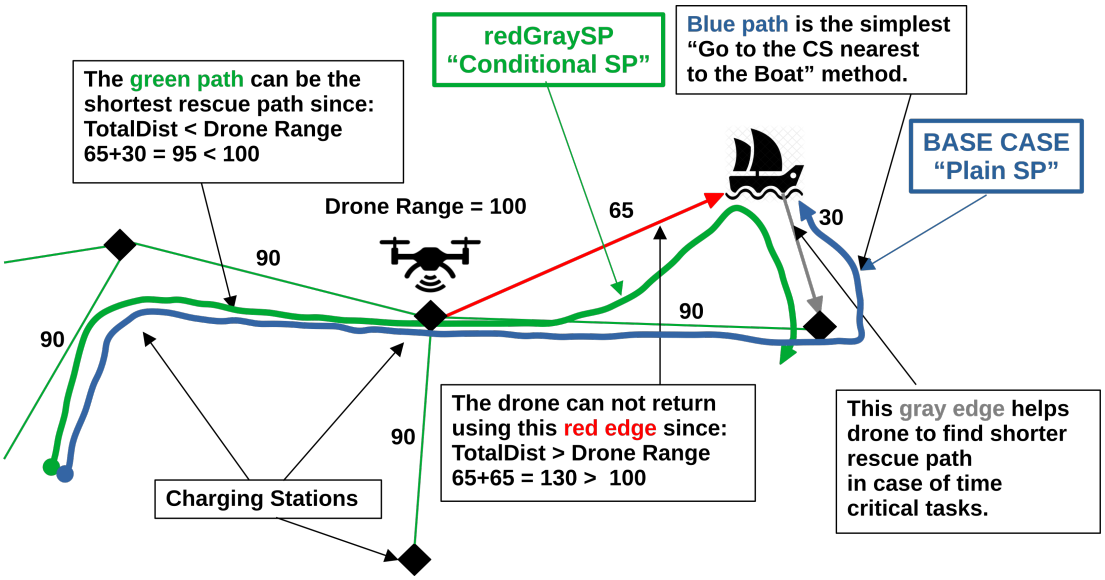
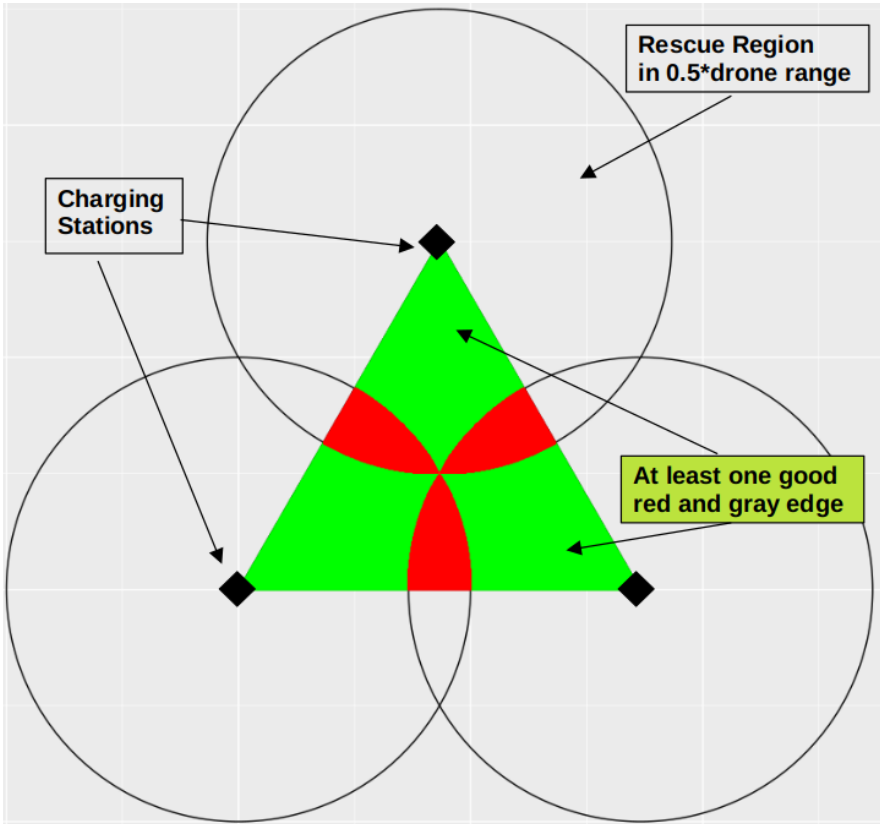


Figure 3.1.15: The good red-gray path making the rescue possible.

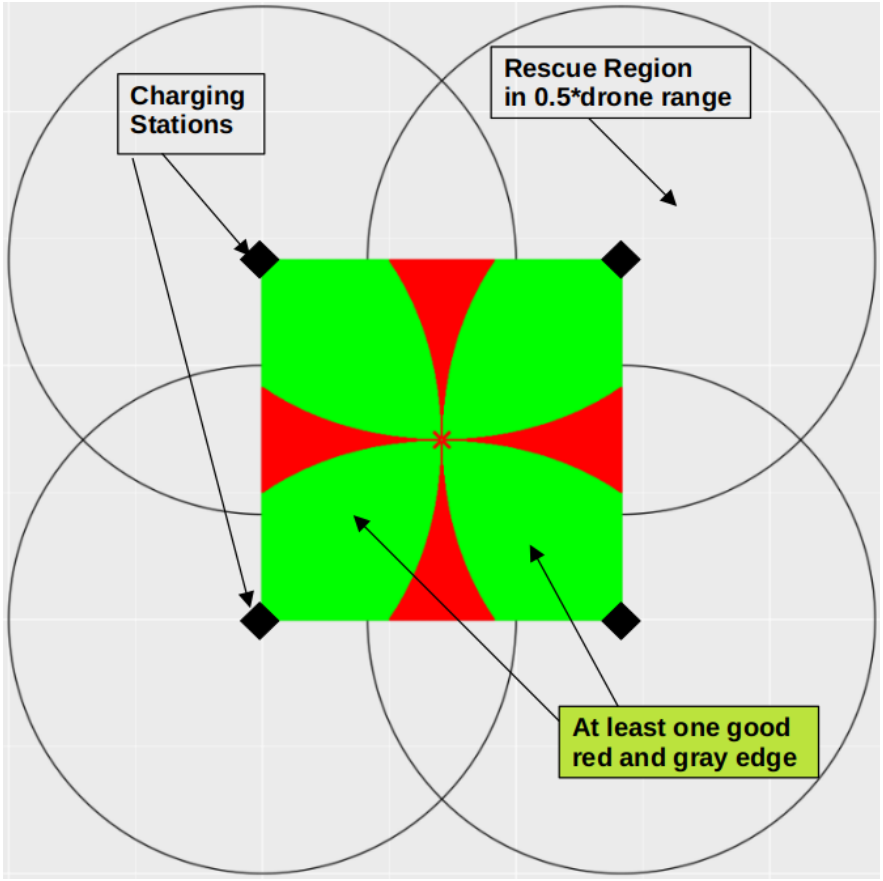
⟨fig:red-gray-edges⟩

The red-gray path edges are dynamically added to the graph structure as the boats make rescue calls. These edges are deleted after the boat is visited by the mission drone. The redGraySP 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 gray 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 gray edge should be considered for the next traversal. If the red edge is used the drone should find and use the gray edge to reach the nearest CS. If it gives a shorter path, the drone can use a red edge after the gray 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-gray edge pair. This analysis can justify the importance of using such a heuristic and can verify the effectiveness of the grid coverage. Figures 3.1.16a and 3.1.16b helps us to understand the analysis focused on the triangular and square regions among different CS grid configurations.



(a) The triangular CS grid.



(b) The square CS grid.

Figure 3.1.16: The prob. of having a "good red-gray path" (the green region) in different CS grid types.

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 associated CS has a gray 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-gray paths" (the sum of distance is less than or equal to drone range) pairs. The red regions represent the locus of the "bad red-gray" edges (when a total distance is greater than the drone range). The probability of having a good red-gray 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.

Table 3.1.1: Point statistics for triangular grid. 69483 points are sampled.

⟨tab:tri-pts⟩

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 3.1.2: Point statistics for square grid. 160801 points are sampled.

⟨tab:sq-pts⟩

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

On Tables 3.1.1 and 3.1.2 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 Gray. "1gR+1G+1bR" means the point that is connected to vertices with 1 good Red edge, 1 Gray 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-gray edge pair. The probability of having at least one good red-gray path is the ratio between the total number of points with at least one good red-gray edge pair 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-gray 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 3.1.1 and 3.1.2 contains some errors.

After finding the probability of having a red-gray 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-gray path should be aligned in a special way in order to benefit from this heuristic. If the drone meets the gray edge first, it will just save the boat and return to BS via that gray edge. But, if the drone meets the red edge first, it will utilize the red-gray path with savings and return to BS. Figure 3.1.17 summarizes all possible directions drone can approach the red-gray edge pair and the outcome of these directions for the triangular grid. A similar analysis can be made for the square grid.

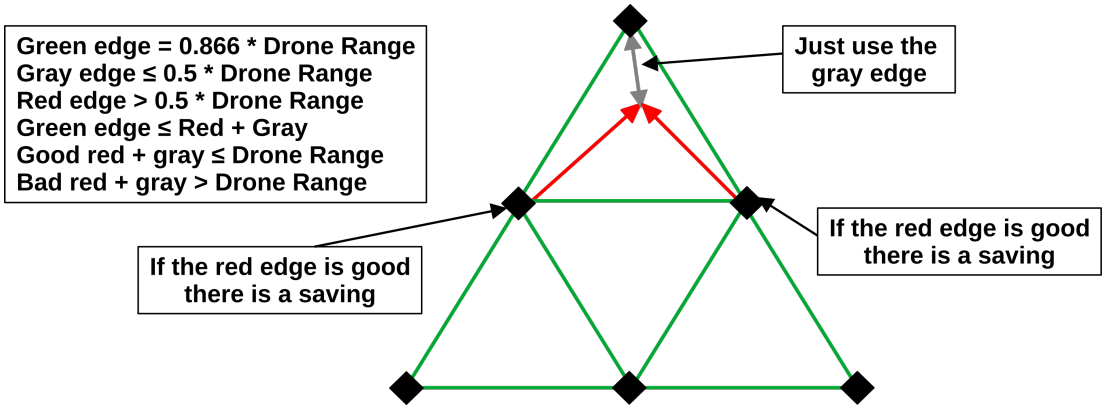


Figure 3.1.17: The coming direction of the drone towards a "good red-gray path" in the triangular CS grid.

-gray-direction⟩

Now we can estimate the probability of benefiting from the redGraySP heuristic by using data from Tables 3.1.1 and 3.1.2. Firstly, we should have a "good direction" (gD in Equations 3.1.1 and 3.1.2) for the boat in which there is a good red-gray path. For the triangular grid Figure 3.1.17 suggests that we can have $\frac{1}{3}$ probability to choose a direction for the drone in the case of the triangular grid when we have a single good Red edge (1gR). In Equation 3.1.1 the probability of benefiting from the red-gray path is given as 0.45. Similar to the triangular grid case, for the square grid the drone can approach the good red-gray edge pair from one of the 4 directions. In Equation 3.1.2 the probability of benefiting from the red-gray path is given as 0.31.

$$\begin{aligned} P(Benefit_{Tri}) &= P(1gR)\times P(gD)+P(2gR)\times P(gD) \\ &= 0.21\times 0.333+0.57\times 0.666 = 0.45 \end{aligned} \tag{3.1.1}$$

?{eq:tri-red-gray-p

$$\begin{aligned} P(Benefit_{Sq}) &= P(1gR)\times P(gD)+P(2gR)\times P(gD) \\ &\quad +P(3gR)\times P(gD) \\ &= 0.3964\times 0.25+0.415\times 0.5 \\ &\quad +0.0046\times 0.75 = 0.31 \end{aligned} \tag{3.1.2}$$

?{eq:sq-red-gray-pr

Finally, we need to know the actual savings from this heuristic. For this we need to consider the edge statistics given on Tables 3.1.3 and 3.1.4.

Table 3.1.3: Edge statistics for triangular grid. 208449 (3 edges * 69483 points) edges are sampled.

⟨tab:tri-edges⟩

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

Table 3.1.4: Edge statistics for square grid. 643204 (4 edges * 160801 points) edges are sampled.

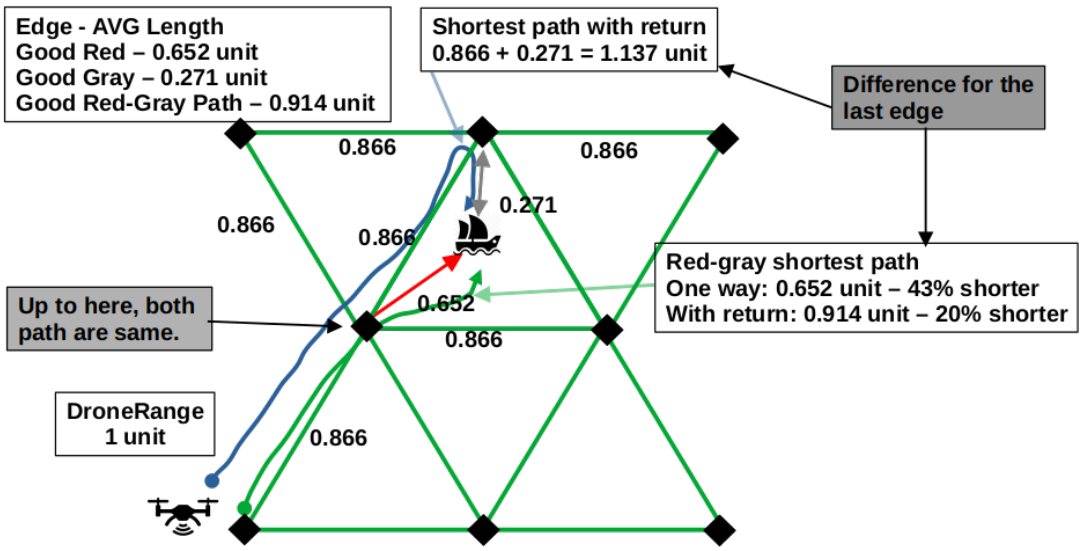
⟨tab:sq-edges⟩

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

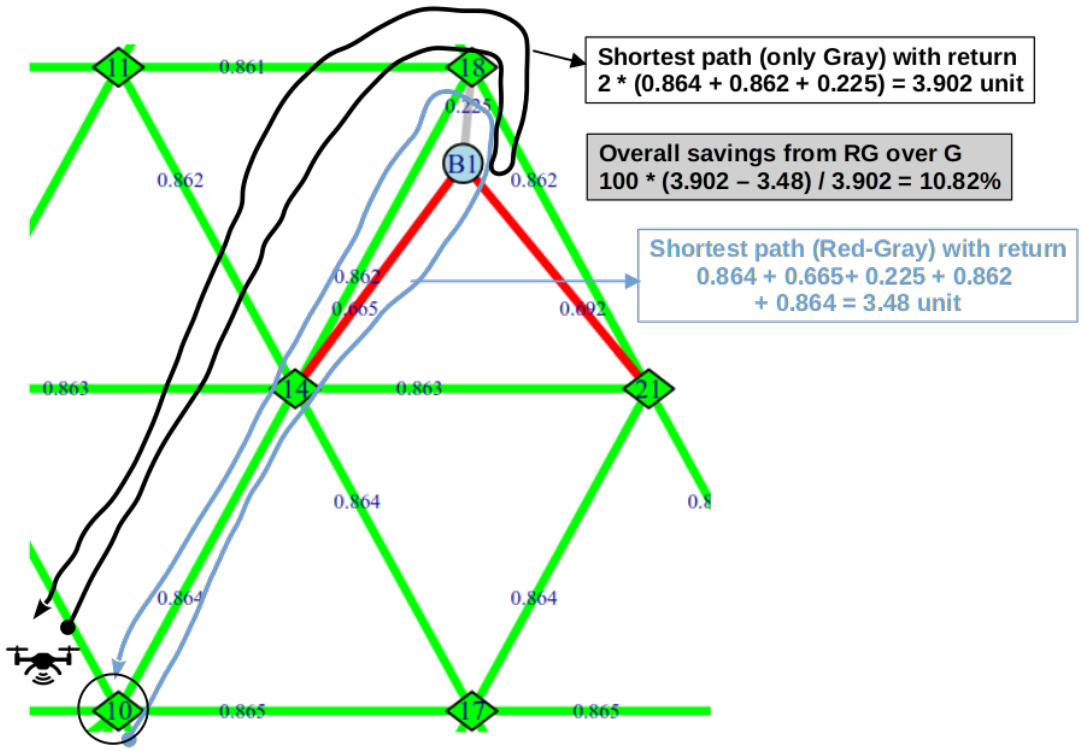
The edge length metrics on these tables are normalized according to the drone range. On tables, "gG" (good Gray) refers to the gray edges that can be part of any red-gray edge pair and "bG" (bad Gray) edge type refers to the gray edges that can not be part of any red-gray edge pair. In short, if a pairing (total edge distance less than nor equal to Drone Range) is possible, both red and gray 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 of the rescue path and present actual figures. The first column, gRG (good Red-Gray Pair), of Table 3.1.3 suggests that when we have a good red-gray 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 gray 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 gray" 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-gray 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 gray 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 3.1.18.



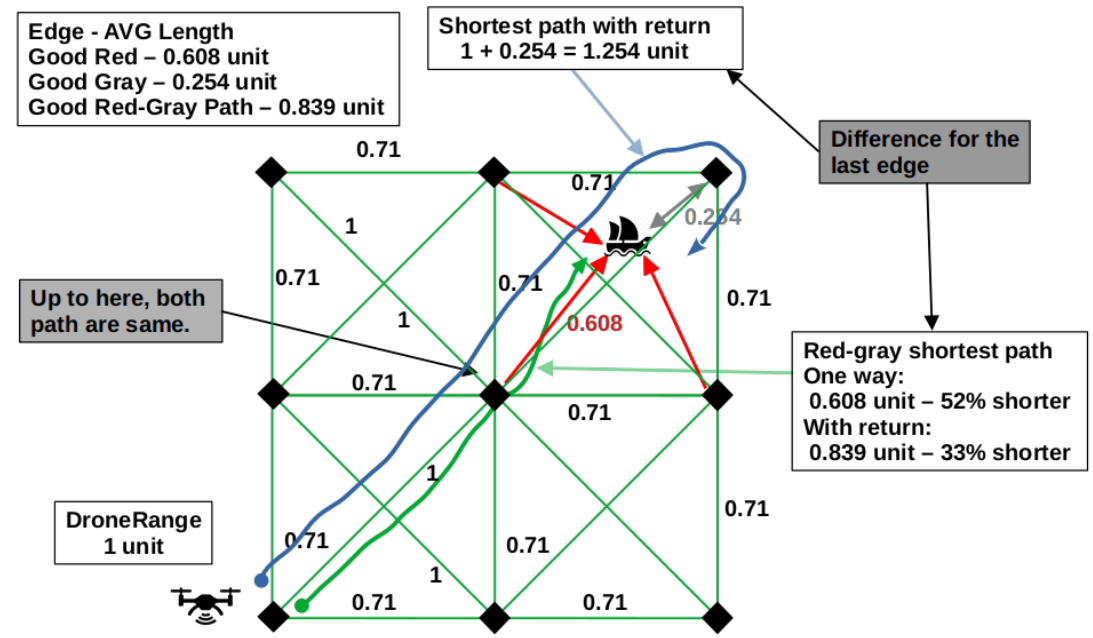
(a) Theoretical savings from red-gray edge heuristic in triangular grid.



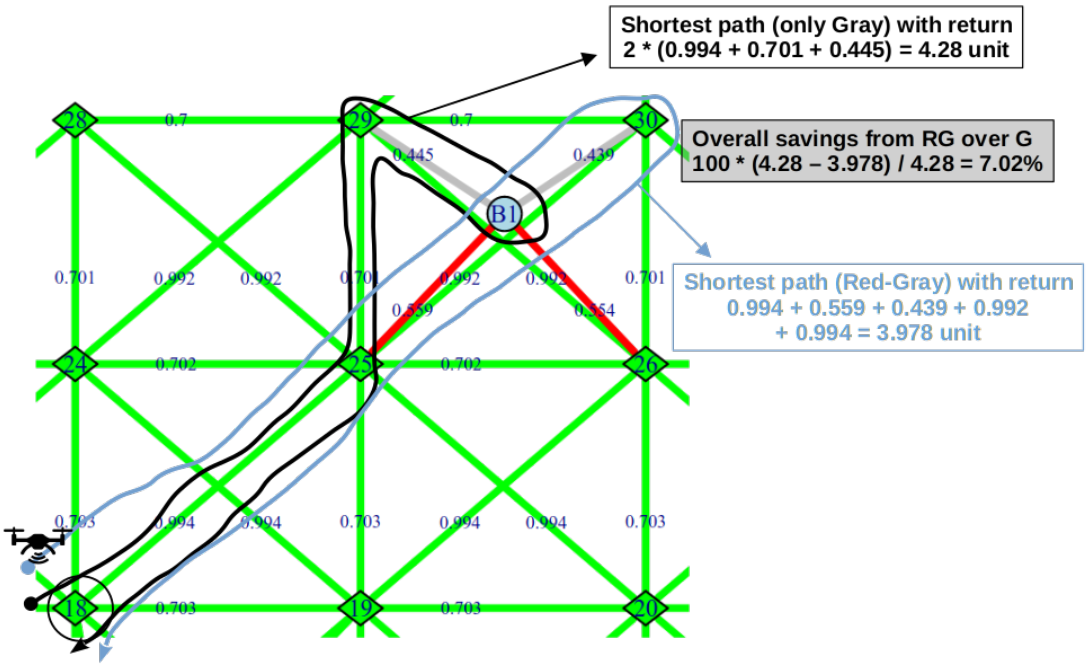
(b) Example savings from red-gray edge heuristic in triangular grid.

Figure 3.1.18: Savings from red-gray edge heuristic for Triangular CS grid configuration.

To find out the actual saving amount for the square grid configuration, Figure 3.1.19 shows an example scenario in which the comparison can be seen between the regular shortest path and the shortest path with the redGraySP heuristics.



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



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

Figure 3.1.19: Savings from red-gray edge heuristic for Square CS grid configuration.

The average length of the "Good Red-Gray Pair" (gRG) is listed as 0.839 units on Table 3.1.4. This edge pair can save us from using the diagonal edge between CSs which has a distance of 1 unit and a gray 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 gray" 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-gray 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 gray 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 3.1.19.

A geometric proof can be given in Figure 3.1.20 to show the red-gray path being better than the base case method in which only the gray edges are considered.

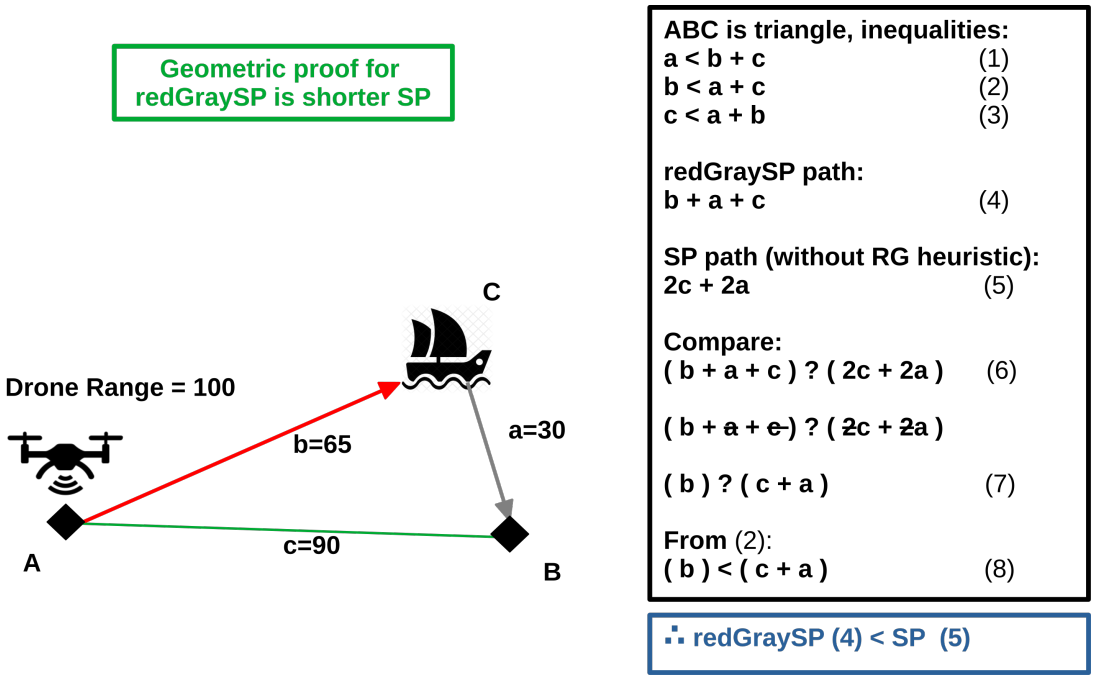
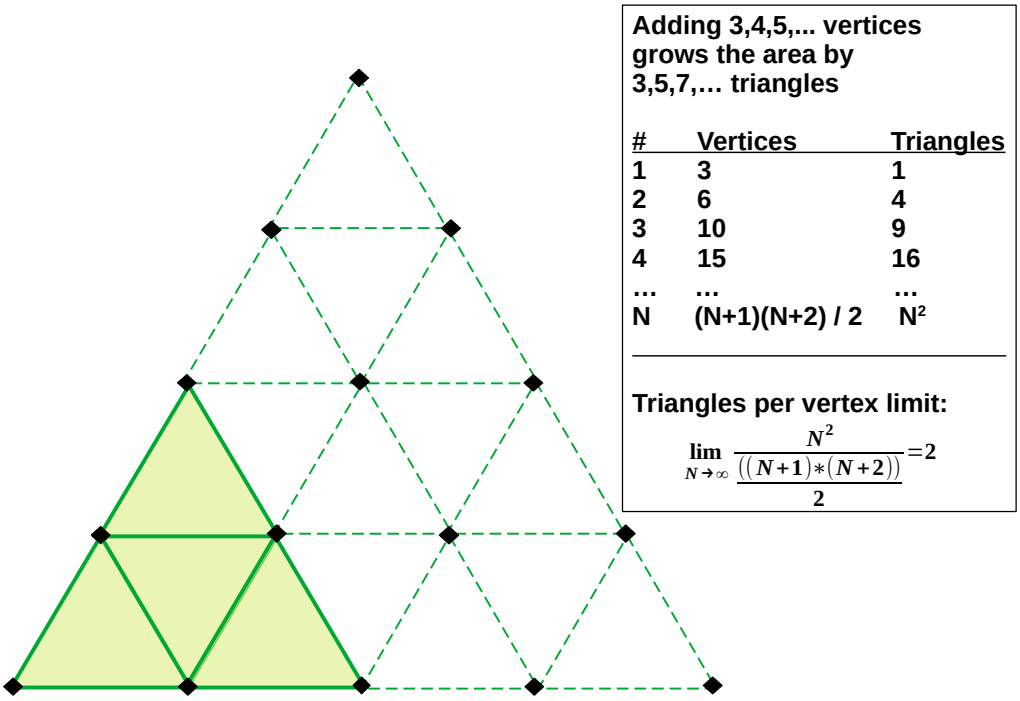


Figure 3.1.20: Geometric proof for the redGraySP.

:red-gray-proof⟩

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 3.1.21a and Figure 3.1.21b by taking the limits when the number of vertices approaches infinity.



Adding 3,4,5,... vertices grows the area by 3,5,7,... triangles

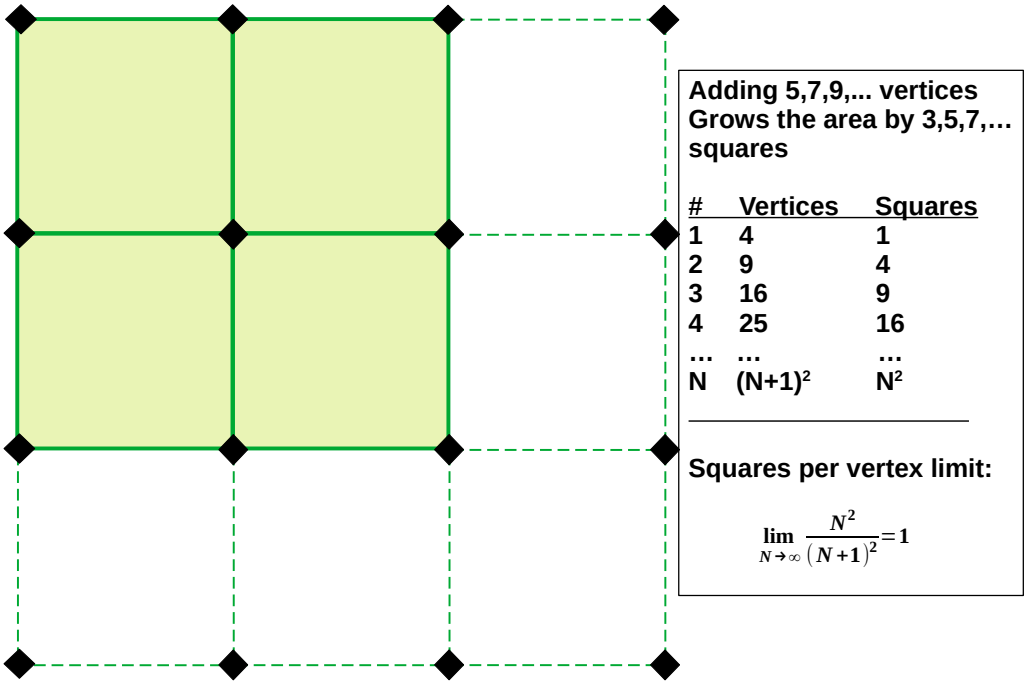
#	Vertices	Triangles
1	3	1
2	6	4
3	10	9
4	15	16
...	...	...
N	(N+1)(N+2) / 2	N^2

Triangles per vertex limit:

$$\lim_{N\to\infty} \frac{N^2}{\frac{((N+1)*(N+2))}{2}} = 2$$

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

sub@fig:tri-limit}?



Adding 5,7,9,... vertices Grows the area by 3,5,7,... squares

#	Vertices	Squares
1	4	1
2	9	4
3	16	9
4	25	16
...	...	...
N	$(N+1)^2$	N^2

Squares per vertex limit:

$$\lim_{N\to\infty} \frac{N^2}{(N+1)^2} = 1$$

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

⟨sub@fig:sq-limit}?

Figure 3.1.21: The ratio of unit area per CS for the triangular and square grid.

?⟨fig:limits⟩?

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 3.1.5a) 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 3.1.5b) 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.

Table 3.1.5: Theoretical comparison of triangular and square grid CS configuration.

:theo-tri-vs-sq⟩

Grid type	AVG SP from BS	Prob. of having a Good Red-Gray Path	Prob. of using a Good Red-Gray 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

On Table 3.1.5 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, is given in drone range units. This path length is specific to the CS grid configurations shown in Figure 3.1.6. While the square grid deployment can provide a better probability of savings and a better amount of savings from the redGraySP 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 3.1.5 the columns **Savings 1-way** and **Savings Return** represent savings just for the last two edges of the optimized path shown in Figures 3.1.18a and in 3.1.19a.

In Figure 3.1.22 the flowchart of the proposed red-gray shortest path algorithm (redGraySP) is shown.

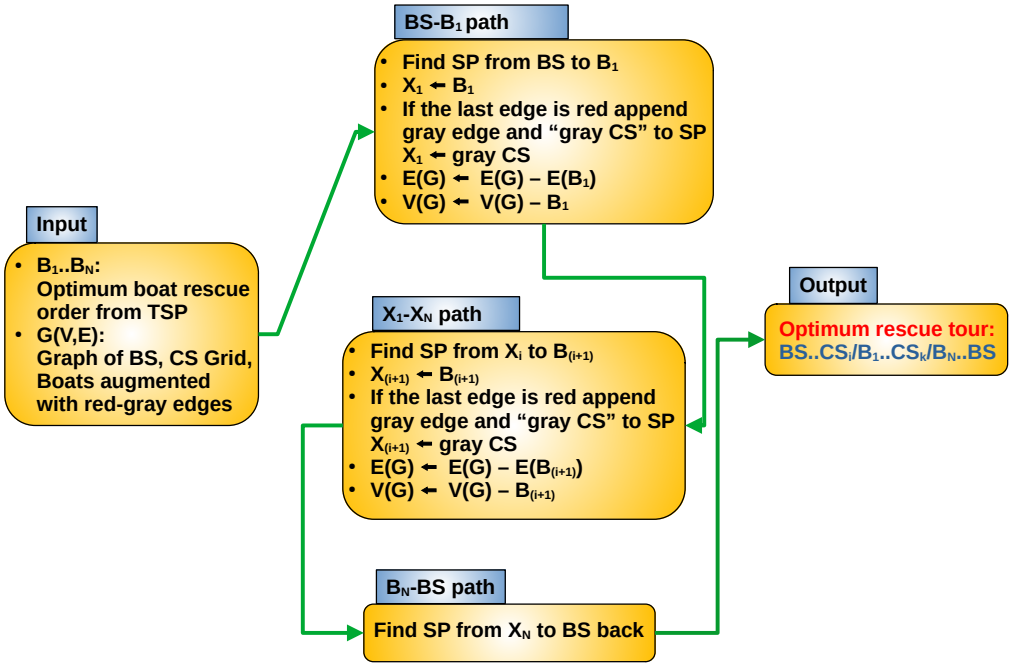


Figure 3.1.22: The flowchart of the proposed red-gray shortest path algorithm (redGraySP).

⟨fig:redgray-algo⟩

In Algorithm 4 the pseudo-code is given for the proposed shortest path algorithm combined with the red-gray heuristic method.

Algorithm 4 The proposed red-gray shortest path algorithm (redGraySP) in pseudo code.

⟨alg:sp-red-gray⟩

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)

The algorithm is given the optimum boat rescue permutation from the proposed TSP heuristic presented in Algorithm 3 as an input along with the graph representation of the BS, CS grid, and boats. This graph is dynamically augmented with red and gray 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 gray edges when it tries to find the shortest path between these boats. As the red-gray 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 gray edges given the vertex id of the boat. The "asVertex()" function creates a vertex instance given the boat id.

Experimental Results and Discussions

⟨ssresults⟩ 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). 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 redGraySP heuristic and pathfinding only with a gray 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-gray and only gray edge versions. For the simulations, we considered the real-life mission region we have presented in Section 3.1.3 in Figure 3.1.6. In the simulations, we wanted to see how much improvement the proposed "redGraySP" can offer for each metric we considered compared to the base case method which was pathfinding with only gray edges.

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) optimization. While the proposed redGraySP 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 3.2.5 details for these big datasets are listed.

Table 3.1.6: Big (1000+ vertices) datasets used in the benchmarks.

tab:dataset-legend⟩

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 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 [127] 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 formalisation of them in the following paragraphs. The TSP tour (or rescue tour) that is presented in Figure 3.2.11 , $\tau = (V_1, \ldots, V_8)$, can be an example for the metrics we explained below.

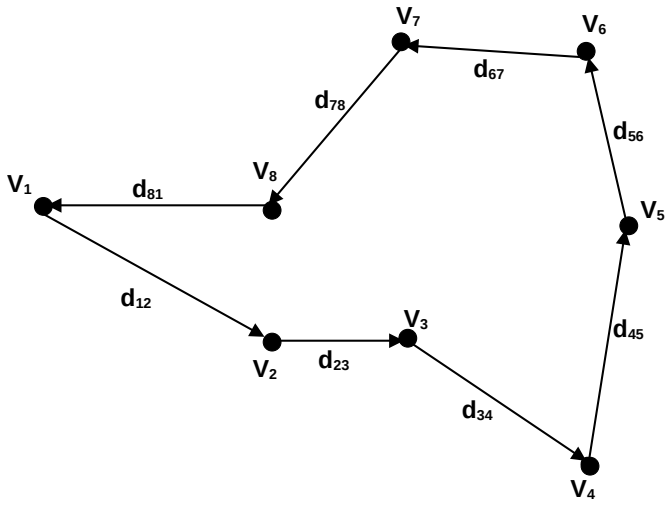


Figure 3.1.23: The TSP on a graph with 8 vertices.

⟨fig:tsp-awt⟩

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 3.2.1 below:

$$\text{Tour Cost}(\tau) \;=\; \sum_{i=1}^{N-1} d_{(i,i+1)} + d_{(N,1)} \qquad (3.1.3)$$?{eq:tour-cost}?

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 3.2.11 below:

$$\text{cyclical AWD}(\tau) \;=\; \frac{1}{N}\Big(\sum_{j=2}^{N}\sum_{i=1}^{j-1} d_{(i,i+1)} + d_{(N,1)}\Big) \qquad (3.1.4)$$?{eq:tsp-cawd}?

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 3.2.12.

$$\text{non cyclical AWD}(\tau) \;=\; \frac{1}{N-1}\Big(\sum_{j=2}^{N}\sum_{i=1}^{j-1} d_{(i,i+1)}\Big) \qquad (3.1.5)$$?{eq:tsp-aawd}?

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 optimization 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 3.1.6.

$$\text{NChargings}(\tau) \;=\; \sum_{i=1}^{N} \left\{ \begin{array}{ll} 1 & \text{if } V_i \text{ is CS} \\ 0 & \text{otherwise} \end{array} \right\} \qquad (3.1.6)$$?{eq:tsp-nc}?

- **Savings:** In order to see improvements that the red-gray 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-gray path heuristic ($Metric_{RG}$) over the base case of using only the gray edges ($Metric_G$). The negative savings mean loss. This can be given by the equation 3.1.7.

$$\text{Savings}\%(Metric) = \frac{100 \times (Metric_G - Metric_{RG})}{Metric_G} \quad (3.1.7)$$ **?**{eq:tsp-savings}**?**

The tour costs and AWD is in meters for the simulations. However, for the big dataset benchmarks, they are in pixel units. The TSP heuristic used in the path-finding framework is very important. The savings can be big with the redGraySP 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.

The simulation results are summarized in tables below (3.1.7, 3.1.8, 3.1.9, 3.1.10, and 3.1.11) 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 (3.1.24a, 3.1.24b, 3.1.25a, 3.1.25b, 3.1.26a, 3.1.26b, 3.1.27a, and 3.1.27b) 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 gray colour. The tour cost savings from the redGraySP 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-gray heuristic over only the gray 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 3.1.12 comparisons are presented for each performance metrics according to the simulation results for different CS grid configurations.

Table 3.1.7: Simulation results for triangular and square grid. Savings are for Red-Gray heuristics over only Gray edge usage. 20 boats randomly generated for 20 simulations. Results are in the form of mean ± standard deviation.

⟨sim20bt20⟩

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

⟨sim20bt40⟩

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

⟨sim20bt60⟩

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

Table 3.1.10: Simulation results for triangular and square grid. Savings are for Red-Gray heuristics over only Gray edge usage. 80 boats randomly generated for 20 simulations. Results are in the form of mean ± standard deviation.

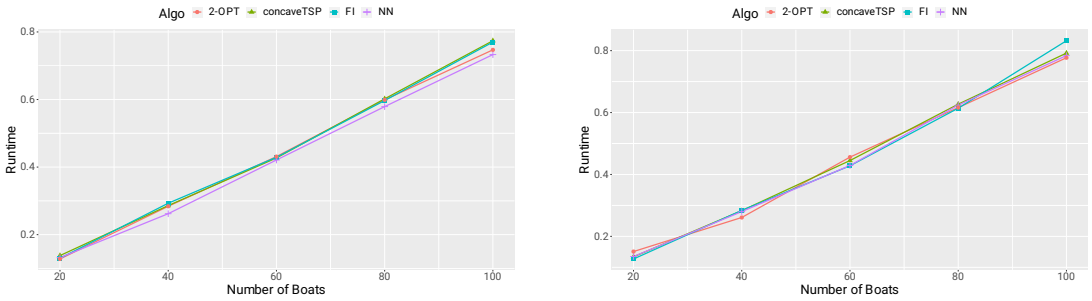
⟨sim20bt80⟩

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

⟨sim20bt100⟩

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



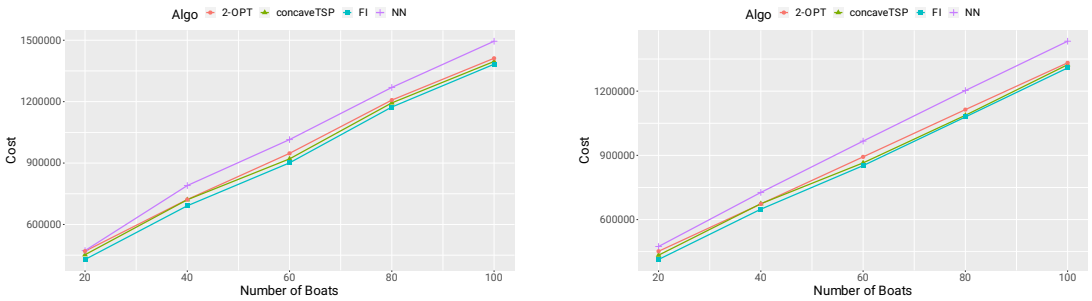
(a) AVG Runtimes for triangular grid.

(b) AVG Runtimes for square grid.

b@figtri-runtime⟩? ?⟨sub@figsq-runtime⟩?

Figure 3.1.24: AVG Runtimes in seconds from 20 sims.

?⟨fig:runtime⟩?



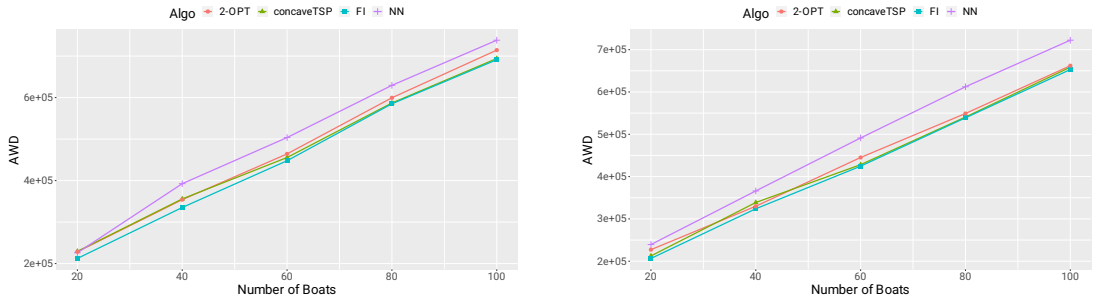
(a) AVG Tour Cost for triangular grid.

(b) AVG Tour Cost for square grid.

?⟨sub@figtri-cost⟩? ?⟨sub@figsq-cost⟩?

Figure 3.1.25: AVG Tour Cost in meters from 20 sims.

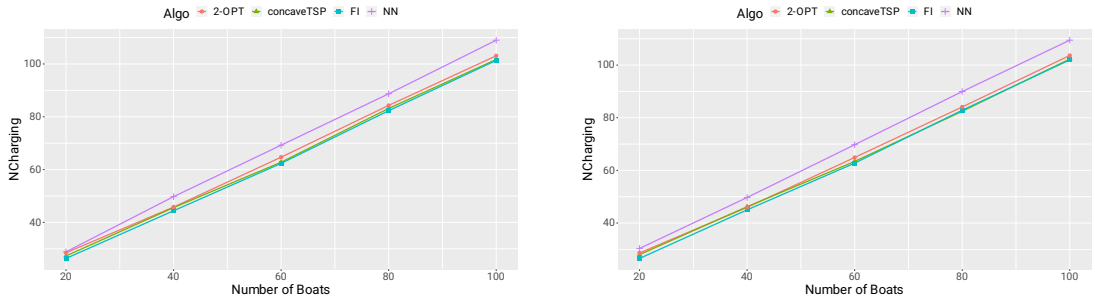
?⟨fig:cost⟩?



(a) AVG AWD for triangular grid.

(b) AVG AWD for square grid.

Figure 3.1.26: AVG AWD in meters from 20 sims.



(a) AVG Number of Chargings for triangular grid.

(b) AVG Number of Chargings for square grid.

Figure 3.1.27: AVG Number of Chargings from 20 sims.

Table 3.1.12: Comparison of triangular and square grid CS configuration according to simulations (Red-gray heuristic from Tables 3.1.7, 3.1.8, 3.1.9, 3.1.10, and 3.1.11).

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)

In Figures 3.1.24a, 3.1.24b, 3.1.25a, 3.1.25b, 3.1.26a, 3.1.26b, 3.1.27a, and 3.1.27b tabular data is presented as plots for triangular and square grids. In these plots, only the algorithms that utilized the red-gray 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 3.1.13, 3.1.14, and 3.1.15 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 gives the best results in every metric we considered in the benchmarks.

Table 3.1.13: Benchmark results for approximate TSP tour costs in units.

⟨tourCost⟩

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 3.1.14: Benchmark results for running time in seconds.

⟨runtime⟩

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

Table 3.1.15: Benchmark results for approximate TSP tour AWD (from vertex 1) costs in units.

⟨awd1⟩

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 redGraySP 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 (3.1.7, 3.1.8, 3.1.9, 3.1.10, and 3.1.11). Looking at the "Cost saving" rows this claim can be verified. In Tables 3.1.13, 3.1.14, and 3.1.15 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.

3.1.4 Conclusions and Future Works

⟨sconc⟩

We tried to address a very specific instance of a novel problem of optimized pathfinding in general drone-based operations assisted with an optimized 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 optimized CS grid and the proposed heuristics.

The optimized CS grid provides complete coverage of the mission region with the min number of CSs and without any blind-spot. In Section 3.1.3 we listed the trade-offs associated with the CS grid configurations we studied.

The proposed redGraySP 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 approxima-

tion 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, "redGraySP", for a generic shortest path algorithm that exploits the geometric regularities of the proposed CS grid.

Currently, the redGraySP 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-optimized 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 [88] 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.

3.2 Novel Concave Hull Based Heuristic Algorithm For TSP

⟨tsp2021⟩ As part of the Pathfinding research, here were present a novel deterministic concave hull-based heuristic algorithm for Euclidean symmetric TSP (Traveling Salesman Problem). The algorithm iteratively creates concentric concave hulls and in a heuristic way merges them into a single tour. We introduced a new metric called the Average Waiting Distance (AWD) of a tour which is an important optimization objective concerning the "cities". Related to AWD, we also introduced another new metric called "min AWD" of a tour which is the minimum AWD when all the "rotations" and "directions" of a tour are considered. In experiments, we saw that the cost-optimum tour known so far does not guarantee optimum AWD or optimum min AWD. Benchmarks are carried out including various standard approximation algorithms by using standard TSPLIB and custom datasets. We performed preliminary statistical analysis on the datasets for classification purposes. The proposed algorithm offered a balanced compromise in performance metrics considered in the benchmarks compared to other algorithms. Especially for the lattice-based (hex) configuration of the cities, the proposed algorithm performed better

in finding the shortest TSP tour with minimum AWD and with shorter running time. We investigated the percentages of the edges in the optimum and in the approximate TSP tours that are coming from the Delaunay Triangulation. Quantitative analysis is carried out, and the savings from the proposed heuristics are tabulated.

3.2.1 Introduction

?⟨intro4⟩?

The TSP is one of the $\mathcal{NP}$-Hard problems that can be reduced from the Hamiltonian Circuit which was among the 21 $\mathcal{NP}$-Complete problems listed by Karp in [79]. The problem has many application areas in operations research, management science, electronics and robotics industry, data analysis in psychology, and, X-Ray crystallography [75, 62, 102]. Basically, in graph-theoretic terms, the problem involves finding the shortest Hamiltonian Cycle given the distance matrix or the coordinates of the vertices or cities. In the article, we used the terms "TSP tour" or simply "tour" for this Hamiltonian Cycle. We also used the terms "vertices" and "cities" interchangeably.

Various approximation algorithms and heuristics were designed to solve the TSP in polynomial time. In Section 3.2.2 we will try to give an overview of them. The essence of the TSP consists of Geometry and Graph Theory. In our work, we tried to use concepts and constructs from both. However, the algorithm we proposed is based on Geometry for the most part. The proposed algorithm is motivated by our previous work in [84]. In the paper, we proposed a framework for sea rescue operations with drones assisted by charging stations configured in a regular geometry on the sea. We analyzed the proposed heuristics for the geometrical configurations of the charging stations we studied. In the last part of the paper, we suggested simple algorithms for finding an optimized path for the type of rescue problem that involved single drone-many boats. In [85], we have benchmarked our TSP implementation in simulations related to the boat rescue case study. This type was a form of TSP in which the drone starts its tour from the base station and after "visiting" all the boats via charging station "hops", returns to the base station. The drone should find the shortest path in rescuing boats. The average waiting time was also one of the constraints of the optimum path. The regular geometric configurations of the charging stations and the consideration of the average waiting time for the optimum tour motivated our efforts in studying the proposed geometry-aware heuristic algorithm for TSP. We also observed that the shortest tours generated by the brute-force method for a small number of boats (5-6) were generally "non-self-crossing loops". This observation led us to consider and investigate convex hull type partial loop heuristics since the utilization of geometry is involved. Most of the optimum TSP tours reported are non-self-crossing space-filling curve type loops. It is also reported in [52] that in the case of Euclidean TSP, the optimum tour does not intersect itself. The research prototype GUI we developed allows us to list the coordinates of the boats that need to be rescued. In this way, the input for the heuristic algorithm can be generated. The initial algorithms were generating concentric convex hulls (rings) and merging them into a single tour using various merging strategies. Generating concentric rings of vertices helps the algorithm for selecting the "nearest neighbors" in the merging process. However sometimes the numbers of "rings" were high and the merging was costly. Then we considered a concave hull instead of a convex hull. The concave hull method gives less number of rings. But for this method the algorithm ([119]) we utilized needs to be run with a proper concavity parameter. The concave hull algorithm we used was a kind of universal-hull algorithm that can be customized with the "concavity parameter". Setting it to higher values makes the algorithm approach to the convex hull algorithm. Lower values give more "rigged" hulls.

We benchmarked the proposed algorithm with standard TSPLIB[1] datasets and custom datasets

[1] http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsp/index.html

to see the results for various performance metrics. The proposed algorithm performed similarly to some of the standard TSP approximation algorithms we listed in the performance section. In grid-based (lattice-based), artificial datasets we created, performed better than the other algorithms. This aspect is important for our previous drone-assisted boat rescue study as we proposed triangular and rectangular grid-based deployment of charging stations for the drones. For many TSP approximation algorithms, the only thing important is the edges (distances) between "cities". However, for the proposed algorithm the geometry is also important as we use the concave hull of the "cities".

We also investigated the proportions of the Delaunay Triangulation (DT) edges in the optimum and approximate tours. The DT edges are duals of the Voronoi Tessellations. Voronoi Tessellations are geometric constructs that divide the region into sub-regions in which any point inside of any sub-region will be closest to the generating point of that region. These generating points are the "cities" or vertices of the Delaunay Triangulation. In this sense, the edges of the DT provide a way to find the nearest neighbors of each city easily. Cases are reported in which the optimum TSP tours contain non-DT edges. However, on the other hand, most of the tours we investigated contain high percentages (over 80%) of DT edges. As it is suggested in many studies, using DT edges for neighbor discovery is a heuristic that should be considered in many applications [75].

We performed a basic exploratory statistical analysis on the datasets we used in the benchmarks by considering the edge distances from the distance matrix. Many standard approximation algorithms are "geometry-blind" in some senses and are focused only on the edge distances. The plain (finding the next min edge every time) Nearest Neighbor algorithm is such an algorithm. Our goal was to see if the datasets can be grouped according to the statistical measures we extracted. The idea was to select different groups of datasets for benchmarking. We tried distribution fitting for the histograms of the edge distances of the datasets. The best-fit distributions are determined and listed. We presented this small exemplary analysis to point out the importance of the dataset selection.

Two tours having the same cost can have a different average waiting time metric. In many time-critical applications, the Average Waiting Time (AWT) of the tour that visits entities is a very important metric. There might be cases that in the overall optimization the AWT can be preferred over the "total tour cost". Rescue operations, industrial applications that heating of certain entities in a specific order is involved, and mechanical hard-disk scheduling can be listed as such applications. Related to this, we proposed the AWD metric and studied its properties. In the benchmarks, this metric is also assessed.

The rest of the paper is structured as follows: In Section 3.2.2 we presented a summary of the previous work related to the topic. The proposed algorithm is explained and various aspects of it are analyzed in Section 3.2.3. In Section 3.2.4 the datasets used in the benchmarks are explained and results are tabulated. Conclusions and future work are listed in Section 3.2.5.

3.2.2 Related Work

⟨related_work4⟩

The TSP literature is very rich since the problem is around for a long time. We like to focus mostly on the approximation algorithms and give an overview of them. The heuristic methods are at the core of the approximation algorithms that try to find a sub-optimum solution in polynomial time. The approximation ratio for the TSP approximation algorithms is the ratio between sub-optimum tour cost and the optimum tour cost. These heuristics are used in various stages of the algorithms. The "initial solution" stage may involve a heuristic in finding a sub-optimum tour in a polynomial time. In the "improvement" stage, the algorithms try to optimize the "initial solution" according to the objectives of the optimization framework, by

using heuristic methods. The works [52] and [29] are two seminal works from which many of the heuristic methods are germinated. The studies [75] and [127] are two excellent works that give rather detailed explanations and analyses on the heuristic methods that are used in the approximation algorithms. The book [62] covered the TSP topic extensively. It presented a chapter on the probabilistic analysis of the TSP and provided reviews of various heuristics. The thesis [51] contains an elaborated chapter (Chapter 2) on the same topic. In these works, the reader can also get an overview of the history of the TSP.

According to [75] the approximation algorithms can be classified into 4 different groups. Namely, construction heuristics, improvement heuristics, special-purpose methods for geometric TSP instances, and the recent approaches centered on the idea of applying stochastic methods in their evolutionary search for the optimum solution. For the construction heuristics, the study lists methods like Nearest Neighbor and the various insertion methods. In the improvement heuristics methods, there are methods like 2-Opt ([48, 122]), in which edge exchanges are applied on the constructed sub-optimum tour for improving the tour cost. For the geometric instances, the study mentions space-filling curve heuristic [15, 120], and methods that use convex hull, Delaunay Graph, and Minimum Spanning Tree type structures for the initial tour. The proposed algorithm enters into this class since it uses a concave hull based initial tour heuristic. In the final, "recent methods" group the study presents the evolutionary methods in which stochastic search is utilized for searching for an optimum solution. The more recent methods like Genetic Algorithm and Simulated Annealing also have that 2-stage structure. Namely the initial solution and the improvement stages. However, they do it in an evolutionary/iterative manner. Various aspects of TSP algorithms based on these methods are studied in papers [96, 142, 153].

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 [57, 47, 2]. 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 [35, 39, 73], 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 [110, 60]. The proposed algorithm 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. Technical aspects related to the utilization of these methods are discussed in detail in Section **??**. The paper [20] presented a method for using the convex hull method in an iterative and exhaustive manner on a given set of points for the construction of many "layers" of hulls. A similar idea is utilized in the proposed algorithm by using the concave hull method.

The DT is a very useful geometric construct, which is the dual of the Voronoi Tessellation (VT). The VT, given the site points, divides the region into sub-regions in a way that any point in any sub-region will always be closest to the associated site point. The sub-region edges form the DT. In the context of TSP, the DT can be used for various purposes. One of the most helpful ways that DT can be used is in finding the nearest neighbor of a given vertex efficiently. The vertices connected by DT edges are the nearest neighbors. The study [41] discussed the $\mathcal{NP}$-Completeness of finding Hamiltonian cycles in DTs. While in the study [91], DT edges are utilized for finding better tours, in the studies [77, 40] counterexamples are given

to show that TSP tours are not always sub-graphs of DTs. In Figure 3.2.1 and in Figure 3.2.2[2] counterexamples are given for the cases in which TSP edges are not always subsets of DT edges. The technical details of such situations are discussed in Section **??**.

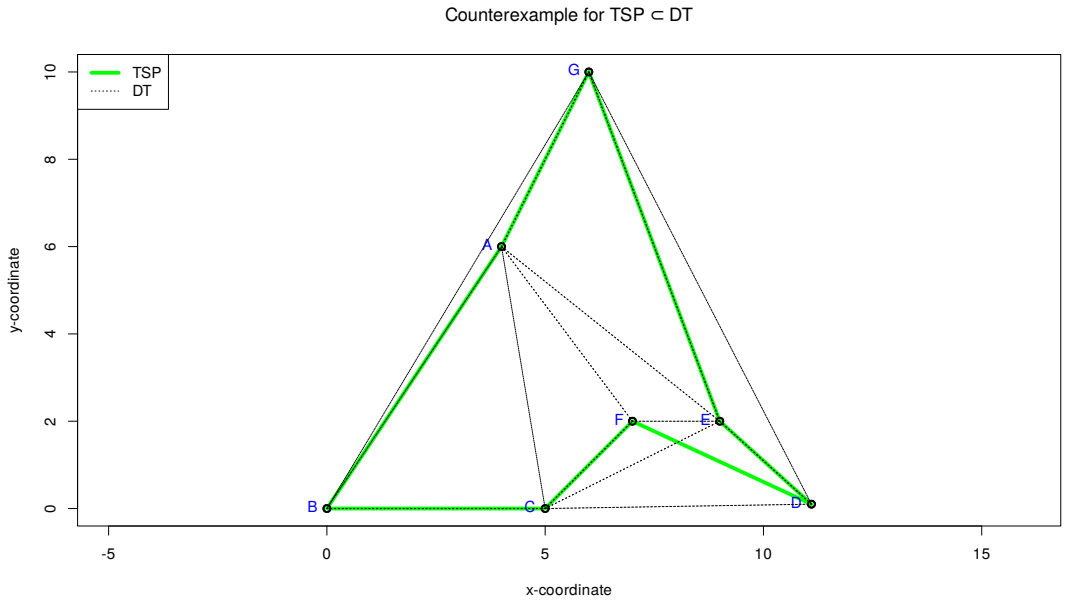


Figure 3.2.1: The counterexample for the claim that TSP edges are always subset of DT edges [40]. The TSP tour is ABCFDEGA. The edge FD is not in DT.

g:counter-vor-7-dt⟩

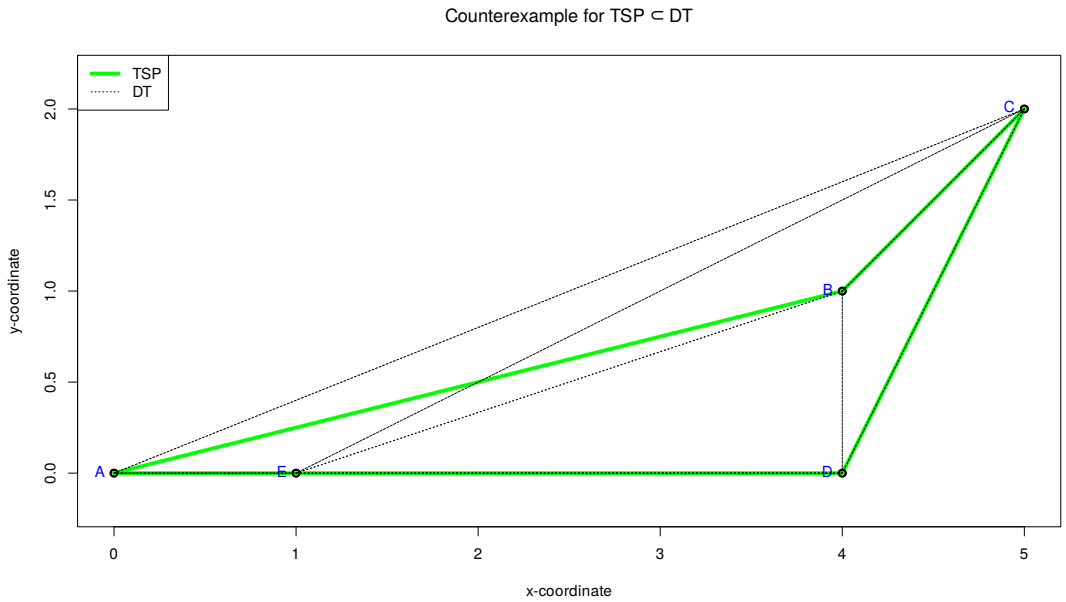


Figure 3.2.2: The counterexample for the claim that TSP edges are always subset of DT edges. The TSP tour is ABCDE. The edge AB is not in DT.

fig:counter-vor-dt⟩

We benchmarked the proposed method and the other standard approximation algorithms

[2]Based on the comments at https://www.ics.uci.edu/~eppstein/junkyard/dt-not-tsp.html

in the context of multi-objective multi-party optimization problem. Multi objective TSP was researched in works like [95], [100], and in [97]. However, in these studies, only generic objectives are assumed.

3.2.3 Proposed Algorithm

⟨algo4⟩ The proposed algorithm is explained in the following sections. First, in Section 3.2.3, an overview related to the motivation of the algorithm is given. The details of its workings are presented on the pseudo-code. After that, in Section 3.2.3, the time and space complexities are discussed. Finally, in Section 3.2.3, an example run is explained with figures tracing the algorithm on a standard TSPLIB dataset.

Overview

⟨algo-overview4⟩ The proposed algorithm can be classified as a hybrid of a geometry-based method with custom "nearest insertion" and "3-Opt-like" (3-edge) heuristics. The algorithm takes the coordinates of the cities (vertices) as input in the format of TSPLIB and produces deterministic output. In the first phase, the proposed algorithm constructs concentric "rings" by using the concave hull method proposed in [119]. For the next phase, the constructed rings are merged (if more than one) based on the nearest insertion and 3-edge heuristics we proposed. We did not try to optimize the space complexity of the proposed algorithm very much. The algorithm uses distance matrix ($\mathcal{O}(n^2)$ space complexity) to make processing faster. The rings are constructed iteratively removing the ring vertices from the initial vertex set till all the vertices are "visited" or 1-2 vertices remain "unvisited". The remaining vertices, if any, are marked and buffered for the next phases of the algorithm.

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 datasets. For the optimum tours, we observed, the overall shape was a "non-self-crossing loop" [52]. 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 neighbor selection process for the merging phase. The leveled 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 neighbor 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. Some memory space can be also be saved instead of searching all the other vertex distances with a bit of computing speed trade-off. The algorithm only needs the distances of the vertices on the related rings. We also experimented with convex hull and alpha hull construction. For the convex hull [14] algorithm some vertices on the edges need to be included after creating hulls which introduces some computational overhead. The number of rings generated was higher with the convex hull algorithm. Decreasing the number of the rings can be achieved to a certain degree by doing extra processing to include the vertices on the edges of the convex hull. But still, the number of rings generated, compared to the concave hull ring generation, is higher. The Delaunay Triangulation degenerates when there are three or more collinear vertices or four or more cocircular vertices. For the alpha hull algorithm, any dataset containing these cases can be a problem since it is a Delaunay Triangulation based method. The Delaunay Triangulation based hull generation algorithms introduce little pertur-

bation to the coordinates to resolve degeneration problems [42, 81]. For the R "alphahull" [16] package we used, the data needed to be normalized into $[0..1]$ interval.

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 neighbor" as the nearest vertex is searched 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 3.2.3. Namely, if the cost of "Pre Dist + Post2Vtx Dist" is less than or equal to the cost of "Post Dist + Pre2Vtx Dist", the path labeled as "A" is chosen. Otherwise, the path labeled 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 3-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. Both heuristics used in the merging phase provide savings.

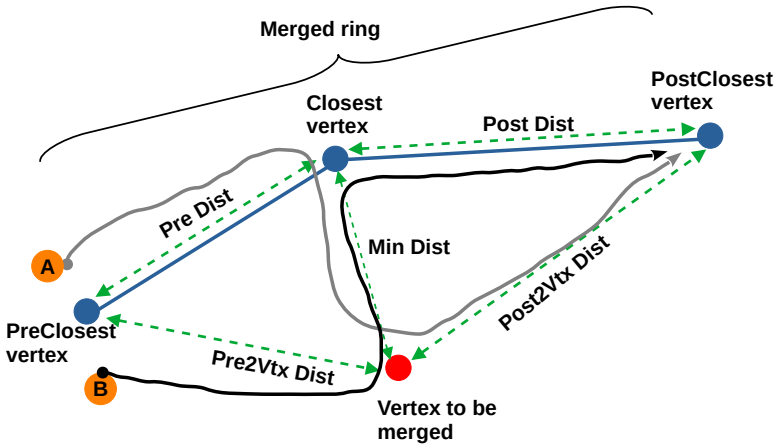


Figure 3.2.3: The 3-edge heuristic used for merging a new vertex to the ring (sub-tour) merged so far.

ge-merge-heuristic⟩

The quantitative results related to the savings are presented in Section **??**. In Algorithm 5 the pseudo-code is given for the proposed method.

Algorithm 5 The proposed algorithm in pseudo code.

⟨alg:merging⟩

Input ▷ VtxList: Euclidean vertex coordinates with numbers: $(i, X_i, Y_i), i = 1 \ldots N$
Output ◁ MergedTour: 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 MergedTour vertex and use 3-edge heuristic

```
NSubTour ← |CHList|
MergedTour ← CHList[1]
if (NSubTour ≥ 2) then
    for each CH ∈ CHList[2..NSubTour] do
        NVtx ← |CH|
        for each Vtx ∈ CH[1..NVtx] do
            MergedTour ← Merge(MergedTour, Vtx)
        end for
    end for
end if
```

Merging RemainedVtx if any

```
if (|RemainedVtx| ≥ 2) then
    MergedTour ← Merge(MergedTour, RemainedVtx)
end if

Return MergedTour
```

Time and space complexity

lgo-complexity4⟩

The proposed algorithm consists of two phases. Namely, the ring construction and the ring merging phases. For the construction phase, we did not add a computationally significant process on top of the method proposed in [119]. This algorithm offers a very flexible hull generation technique that can be adjusted with the "concavity" parameter which is a measure for concaveness. The higher the value of this parameter, the more similar the resulting hull becomes to the convex hull. As the value of the concavity parameter decreases, the algorithm generates a more detailed (rigged) hull of the given vertex set. For our purposes, we set the value of the concavity as 0.9. We invite interested readers to further research the issue of finding "optimum concavity" value for obtaining better tour cost. For this purpose, some kind of geometric configuration measure might be needed considering the edge distances. According to the paper [119], 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 optimized 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. Currently, in our implementation, every "inner ring" (ring to be merged) vertex is merged. The decision for merging or not merging can be further elaborated by adding a "heuristic vertex buffering" scheme in which not only the individual vertices are considered but also a chain of them can be merged depending on the local cost minimization. So in general the proposed method is bounded by $\mathcal{O}(kN\log N)$ time complexity. In Table 3.2.1 the proposed algorithms (CH: with heuristic, CNH: without heuristic) are compared to different time complexity growth rates ($\mathcal{O}(N)$, $\mathcal{O}(N\log N)$, and $\mathcal{O}(N^2)$). The time overhead of the proposed 3-edge heuristics is negligible when we compare the columns of the proposed methods. Table 3.2.2 lists the time complexities of the algorithms used in benchmarks. We did not include the distance matrix generation time in Table 3.2.1 and in Figure 3.2.4. The analysis is focused on the proposed heuristics. The brute-force distance creation part can be optimized in many ways regarding space and time. In Figure 3.2.4 the quantitative data is plotted for a better view of the comparison. The extreme values in Table 3.2.1 are ignored for better scaling of the plot. The reader can find the details of the datasets listed in Section **??**.

Table 3.2.1: Running times (secs) for the proposed algorithms compared to different growth rates.

untime-proposed⟩

DataSet	N	CH	CNH	$\mathcal{O}(N)$	$\mathcal{O}(N \log N)$	$\mathcal{O}(N^2)$
att48	48	0.018	0.018	0.018	0.018	0.018
berlin52	52	0.024	0.023	0.020	0.002	0.021
pr76	76	0.027	0.025	0.029	0.019	0.045
kroA100	100	0.029	0.028	0.038	0.040	0.078
lin105	105	0.030	0.030	0.039	0.044	0.086
ch130	130	0.034	0.033	0.049	0.070	0.132
ch150	150	0.037	0.037	0.056	0.092	0.176
a280	280	0.070	0.072	0.105	0.267	0.613
pcb442	442	0.105	0.104	0.166	0.531	1.526
pr1002	1002	0.179	0.168	0.376	1.647	7.844
pr2392	2392	0.493	0.465	0.897	5.058	44.701
rl5915	5915	1.666	1.669	2.218	15.405	273.338
usa13509	13509	6.575	6.489	5.066	41.219	1425.727
pla33810E	33810	33.736	33.271	12.679	119.944	8930.595

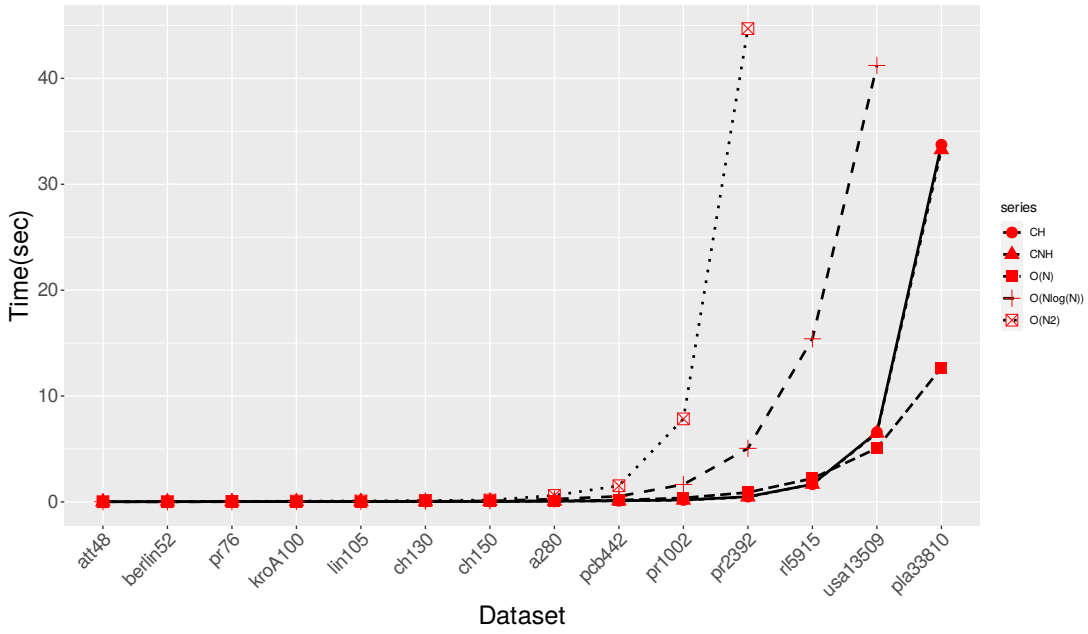


Figure 3.2.4: The time complexity of the proposed algorithm.

⟨fig:rt-algo⟩

Table 3.2.2: The time complexities of the standard algorithms used in benchmarks [75, 127].

⟨tab:std-runtime⟩

Algorithm	Time Complexity
Nearest Neighbor	$\mathcal{O}(N^2)$
Repetitive Nearest Neighbor (no parallelism)	$\mathcal{O}(N^3)$
Nearest Insertion	$\mathcal{O}(N^2)$
Cheapest Insertion	$\mathcal{O}(N^2 \times log(N))$
Furthest Insertion	$\mathcal{O}(N^2)$
Arbitrary Insertion	$\mathcal{O}(N^2)$
2-Opt	$\mathcal{O}(N^2)$

For the space complexity, we did not consider any optimization. A distance matrix with space complexity of $\mathcal{O}(N^2)$ is created and used whenever it is necessary to get the distances. In several TSPLIB datasets, the distance matrix is previously calculated and is given in the input file. For this reason, the preparation of the distance matrix is not considered in the time complexity. The reader also should be aware of the possibility of creating this matrix in a very speedy manner by using parallelism. For small datasets, the distance matrix creation is less than 0.001 secs. We used an R function to create the distance matrix. Whether some kind of parallelism is used or not we did not investigate. However, for big datasets (number of vertices > 1000), the matrix creation time increases very much. Table 3.2.3 shows the space and time measures for the distance matrix creation regarding the big datasets. For the distance matrix, each element is a 64-bit floating-point number. The paper [17] contains valuable technical discussions on the data structures and time/space complexities of TSP algorithms.

Table 3.2.3: The space and time complexity of the distance matrix generation for big datasets.

⟨tab:big-dstmtx⟩

DataSet	N	N^2	Memory	Time (secs)
pr1002	1002	1004004	8.2 MB	0.016
pr2392	2392	5721664	46.1 MB	0.352
rl5915	5915	34987225	280.7 MB	1.078
usa13509	13509	182493081	1.5 GB	3.303
pla33810E	33810	1143116100	9.1 GB	16.715
Custom lattices 10000	10000	100000000	801.3 MB	1.831
Custom lattices 20000	20000	400000000	3.2 GB	5.692
Custom lattices 30000	30000	900000000	7.2 GB	12.116

Example run

⟨algo-run4⟩

For a better understanding of the working of the proposed algorithm, we presented several graph plots that show the various stages of the algorithm. Figure 3.2.5 shows the standard TSPLIB dataset “pr76” with 76 vertices labeled with red color.

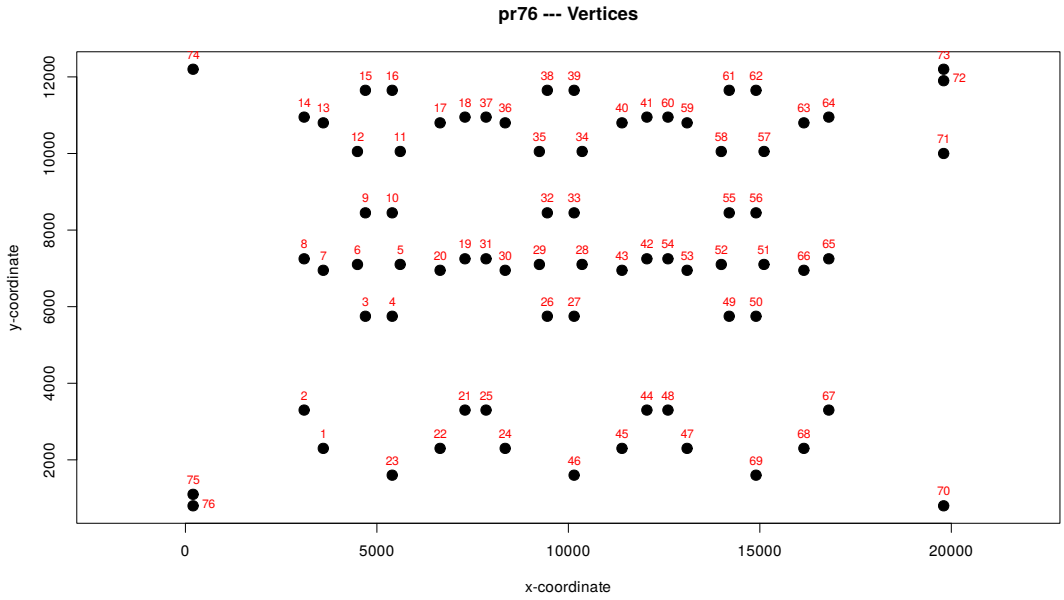


Figure 3.2.5: "pr76" TSPLIB dataset.

⟨fig:pr76-vtx⟩

Figure 3.2.6 shows the optimum tour of the TSPLIB dataset "pr76" with the Delaunay Triangulation edges overlapped. We can see that only two edges of the optimum tour are not from DT edges.

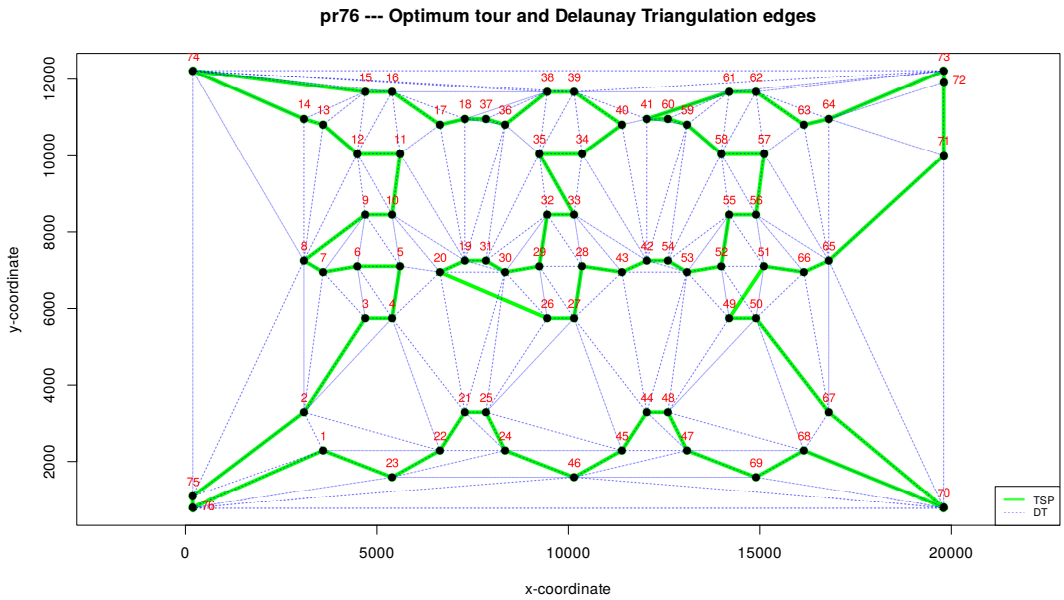


Figure 3.2.6: "pr76" optimum tour (green) with Delaunay Triangulation edges (blue dashed).

⟨fig:pr76-dt⟩

Figure 3.2.7 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 color is the first ring and the inner ring with green color is the second ring.

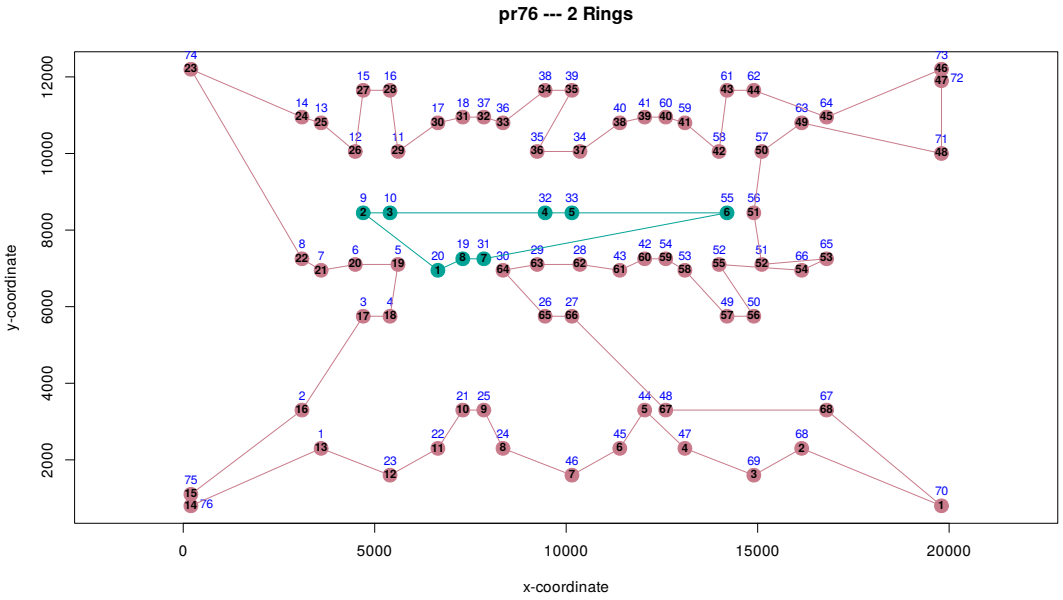


Figure 3.2.7: Rings generated by concave hull for pr76.

⟨fig:pr76-rings⟩

Figure 3.2.8 and 3.2.9 show the process of merging the first and the last point from the second ring respectively. In Figure 3.2.8 vertex number 20, the first vertex of the second ring is paired with the vertex number 5 (ring 1 vertex 19, colored 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 colored with red.

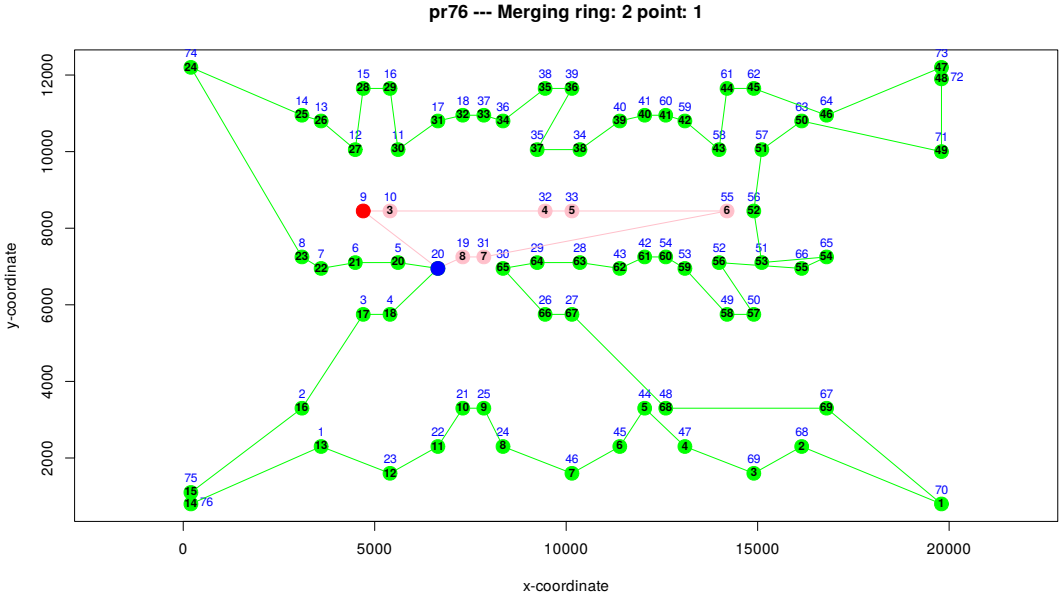


Figure 3.2.8: Merging the first point from the second ring. The next ring marked with red color.

⟨fig:pr76-p1⟩

In the last merging round vertex number 19 (the last vertex of the second ring) is merged to vertex number 31 which was previously merged to ring 1 (Figure 3.2.9).

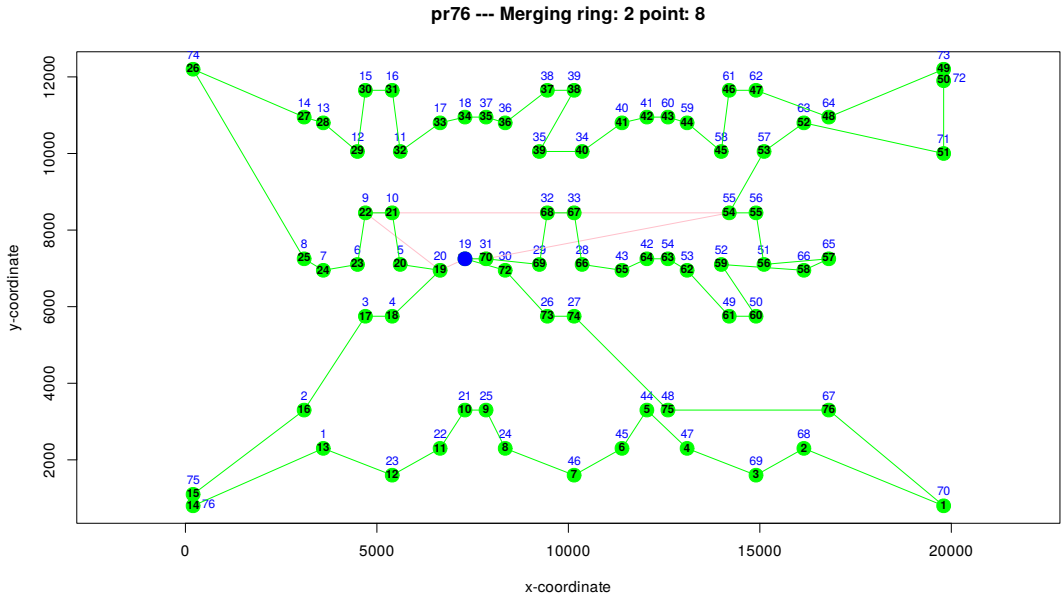


Figure 3.2.9: Merging the last point from the second ring.

⟨fig:pr76-p8⟩

Figure 3.2.10 shows the final "merged ring" (the approximate tour from the proposed algorithm) in green color. 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. For the standard TSPLIB dataset pr76, the proposed algorithm generated the best approximation compared to the other algorithms benchmarked (Section **??**).

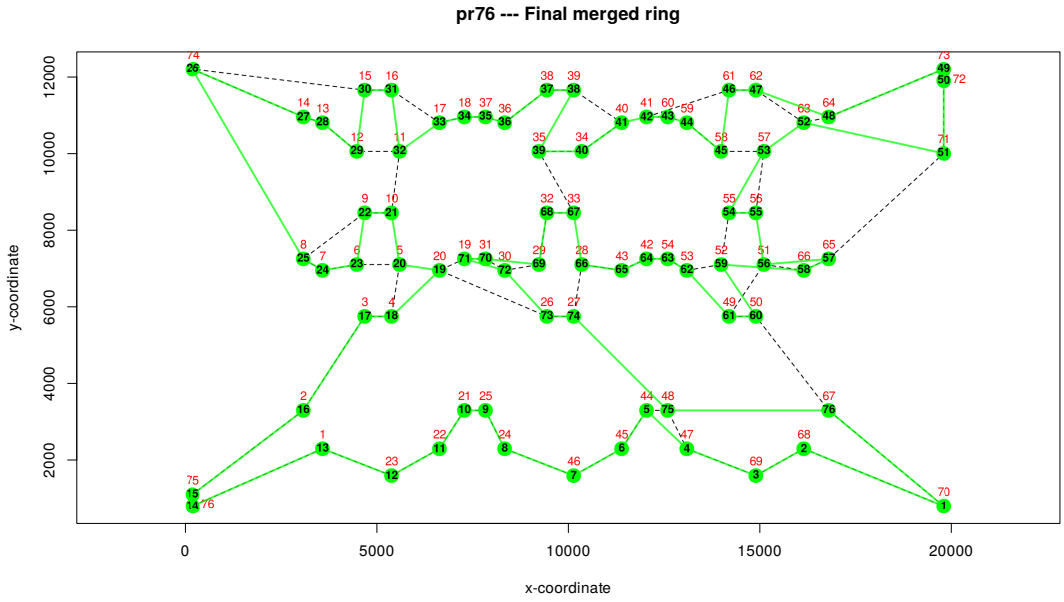


Figure 3.2.10: The final merged ring and the optimum tour for pr76.

⟨fig:pr76-final⟩

3.2.4 Benchmark Metrics and Performance Results

⟨perf4⟩ After we implemented the proposed algorithm in R language we benchmarked it against several standard approximation algorithms by using standard datasets and several custom datasets. We performed statistical analysis to see if different classes of datasets are selected. For benchmarking various metrics are selected by considering a general multi-party multi-objective optimization problem. In the case of TSP, the parties can be named as the salesman and the cities. Objectives can be minimum length tor for the salesman and the minimum average waiting time for the cities. Generally in TSP literature, the "tour cost" in less running time is the only metric that is researched. However, in our work, we included several other metrics and presented discussions about them. In some optimization problems, multiple parties may be involved. In fact, in the TSP, "cities" are the second party in the overall optimization. While the tour cost is mainly important for the salesman, cities may request lower waiting times. For certain shortest TSP tours, the famous "convoy effect" may be so displeasing for the cities. In that sense, we considered AWT (AWD in our study) as one of the metrics that should be investigated. Concerning that metric, just to see the potential of the TSP tour that is found, "min AWD" is presented and included in the benchmarks. We also think collecting some data on the percentage of the tour edges that are also DT edges could be interesting. So overall, we collected data for six different metrics: Approximation Ratio, Running Time, Approximate Tour Cost, DT edge Percentage, AWD, and min AWD.

The details on the metrics are presented in Section 3.2.4. In Section 3.2.4 we listed the datasets and we tabulated the quantitative data gathered from the benchmarks. We suggested the idea of performing a statistical analysis to see if there is any relationship between various statistical measures specific to datasets and the algorithm performance. For this, we presented an example analysis for the Nearest Neighbor algorithm. Then we tried to quantify the savings the proposed heuristics can provide. Finally, the comparative results from the benchmarking of the algorithms are tabulated.

Metrics

⟨metrics4⟩ The Approximation Ratio is the ratio between the approximate tour cost that the approximation algorithm found and the optimum tour cost known so far. Most of the TSPLIB datasets have the cost and the tour vertex permutation given in a separate data file. For some datasets, there is no optimum tour cost. For these datasets, we omitted this metric. Also for the custom datasets we generated, we had neither the tour cost nor the optimum tour itself. The approximate tour costs as the total edge distance between the tour vertices are reported separately too. The running time is measured as the CPU time reported from the R "tictoc" package [72]. The standard approximation algorithms like nearest insertion or 2-Opt are used from the R package "TSP" [64]. For the proposed algorithm we used R language. The utilization of different dependent R packages for our algorithm and for the other algorithms and the differences in the used R data structures that has the same function (matrix vs dataframe in R) makes it hard to compare these algorithms based on the running time. However, as an experiment, we measured and reported the running times. We thought that reporting the percentage of the optimum/approximate tour edges that are also DT edges would be interesting for comparison purposes.

The AWD and min AWD were the novel metrics we proposed and measured. These metrics are important in multi-party multi-objective optimization problems. To explain these metrics, we tried to give a mathematical formalization of them in the following paragraphs.

Figure 3.2.11 shows an example TSP tour, $\tau = (V_1, \ldots, V_8)$, on the graph of 8 vertices.

The total "tour cost" (total distance traveled by the salesman) is the sum of all the edge

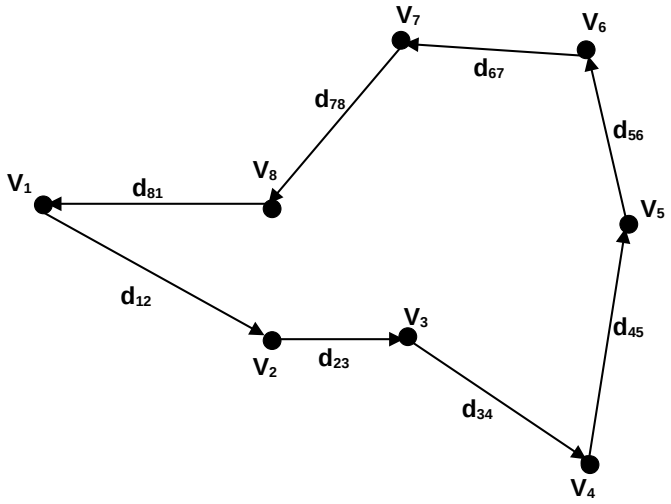


Figure 3.2.11: The TSP on a graph with 8 vertices.

⟨fig:tsp-awt⟩

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_1 is the starting and ending vertex of the tour. More specifically, if the vertices of the tour are labeled from $V_1, \ldots, V_N$, the tour cost (cyclical) can be given with the following equation 3.2.1 below:

$$\text{Tour Cost}(\tau) \;=\; \sum_{i=1}^{N-1} d_{(i,i+1)} + d_{(N,1)} \qquad (3.2.1)$$?{eq:tour-cost}?

Now we define the directional (initially given/forward, opposite/backward) rotation of a tour which will help us for the analysis of the metrics related to the average waiting time of the vertices on the optimum or approximate optimum TSP tour. Let τ be the TSP tour with N vertices specified by the vertex sequence $V_1, \ldots, V_N$. For any TSP tour $\tau = (V_1, \ldots, V_N)$ the forward (F) rotation of a single position (which is another tour) can be stated as in the equation 3.2.2 below:

$$\rho_1^F(\tau) \;=\; (V_2, \ldots, V_N, V_1) \qquad (3.2.2)$$?{eq:tsp-rotF}?

Similarly, the backward (B) rotation of a single position can be stated as in the equation 3.2.3 below:

$$\rho_1^B(\tau) \;=\; (V_N, V_{N-1}, \ldots, V_1) \qquad (3.2.3)$$?{eq:tsp-rotB}?

The equations 3.2.4, 3.2.5, and 3.2.6 below should be noted about the rotation operations we defined for TSP tour $\tau = (V_1, \ldots, V_N)$.

$$\rho_0^F(\tau) = \rho_N^F(\tau) = \tau \qquad (3.2.4)$$?{eq:rot0F}?

$$\rho_0^B(\tau) = \rho_N^B(\tau) = (V_1, V_N, V_{N-1}, \ldots, V_2) \qquad (3.2.5)$$?{eq:rot0B}?

$$\rho_1^B(\tau) = \tau^{-1} = (V_N, V_{N-1}, \ldots, V_1) \qquad (3.2.6)$$?{eq:rot1B}?

So in general, for a TSP tour τ, the directional rotation of $k (0 \leq k \leq N-1 \; or \; k = k \mod N)$ positions can be given as in the equations 3.2.7 and 3.2.8 below:

$$\rho_k^F(\tau) = (V_{((1+k-1) \bmod N)+1}, V_{((2+k-1) \bmod N)+1}, \cdots, V_{(((N-1)+k-1) \bmod N)+1}, V_{((N+k-1) \bmod N)+1}) \qquad (3.2.7)$$?{eq:tsp-rotkf}?

$$\rho_k^B(\tau) = (V_{((N-(1+k-1)) \bmod N)+1}, V_{((N-(2+k-1)) \bmod N)+1}, \cdots, V_{((N-((N-1)+k-1)) \bmod N)+1}, V_{((N-(N+k-1)) \bmod N)+1}) \quad (3.2.8)$$?{eq:tsp-rotkb}?

Example rotations are given below in 3.2.9 and 3.2.10 for a TSP tour $\tau = (V_1, V_2, V_3, V_4, V_5, V_6, V_7)$ with $N = 7$.

$$\begin{aligned}
\rho_0^F(\tau) &= (V_1, V_2, V_3, V_4, V_5, V_6, V_7) \\
\rho_1^F(\tau) &= (V_2, V_3, V_4, V_5, V_6, V_7, V_1) \\
\rho_2^F(\tau) &= (V_3, V_4, V_5, V_6, V_7, V_1, V_2) \\
\rho_3^F(\tau) &= (V_4, V_5, V_6, V_7, V_1, V_2, V_3) \\
\rho_4^F(\tau) &= (V_5, V_6, V_7, V_1, V_2, V_3, V_4) \\
\rho_5^F(\tau) &= (V_6, V_7, V_1, V_2, V_3, V_4, V_5) \\
\rho_6^F(\tau) &= (V_7, V_1, V_2, V_3, V_4, V_5, V_6)
\end{aligned} \quad (3.2.9)$$?{eq:ex-rotkf}?

$$\begin{aligned}
\rho_0^B(\tau) &= (V_1, V_7, V_6, V_5, V_4, V_3, V_2) \\
\rho_1^B(\tau) &= (V_7, V_6, V_5, V_4, V_3, V_2, V_1) \\
\rho_2^B(\tau) &= (V_6, V_5, V_4, V_3, V_2, V_1, V_7) \\
\rho_3^B(\tau) &= (V_5, V_4, V_3, V_2, V_1, V_7, V_6) \\
\rho_4^B(\tau) &= (V_4, V_3, V_2, V_1, V_7, V_6, V_5) \\
\rho_5^B(\tau) &= (V_3, V_2, V_1, V_7, V_6, V_5, V_4) \\
\rho_6^B(\tau) &= (V_2, V_1, V_7, V_6, V_5, V_4, V_3)
\end{aligned} \quad (3.2.10)$$?{eq:ex-rotkb}?

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 traveling direction. This cost is in the interest of the salesman's party. Another cost that can be estimated for the TSP tour is the Average Waiting Time (AWT), which is in the interest of the customers' party. We assumed that the speed of the salesman 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) 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, for the last vertex or customer we assume that a "ghost vertex or customer" is waiting for the arrival of the salesman at the first vertex of the tour. In this sense, the "cyclical AWD" is estimated. The cyclical AWD of tour τ which is given in the equation 3.2.11 below:

$$\text{cyclical - AWD}(\tau) \;=\; \frac{1}{N}\Big(\sum_{j=2}^{N}\sum_{i=1}^{j-1} d_{(i,i+1)} + d_{(N,1)}\Big) \quad (3.2.11)$$?{eq:tsp-cawd}?

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 3.2.12.

$$\text{non-cyclical - AWD}(\tau) \;=\; \frac{1}{N-1}\Big(\sum_{j=2}^{N}\sum_{i=1}^{j-1} d_{(i,i+1)}\Big) \qquad (3.2.12)$$?{eq:tsp-aawd}?

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. For example, the optimum tour (starting from vertex no 1) for the TSPLIB dataset "berlin52" [3] gives AWD as 4131.786 (non-cyclical AWD is 4064.873). However, for the same dataset, the approximate tour (starting from vertex no 1) from the "repetitive_nn" algorithm gives AWD as 3961.128 (non-cyclical AWD is 3878.362). We can define the "min AWD" of a tour as the smallest AWD value of all its rotations (including all tours that starts from the different vertices of the tour other than the vertex number 1) and directions (both forward and backward). To find min AWD, the cost should be calculated starting from each vertex on the tour for both directions. The formula for AWD is given in the equation 3.2.13 below:

$$\text{min AWD}(\tau) = min(min(\rho^{F}_{i=1,\ldots,N}(\tau)), min(\rho^{B}_{i=1,\ldots,N}(\tau))) \qquad (3.2.13)$$?{eq:tsp-minawd}?

The optimum tours (starting from vertex no 1) given with the TSPLIB datasets do not give the smallest "min AWD" value. For example, the dataset "att48" from the TSPLIB gives an AWD value of 15623.1 (non-cyclical AWD is 15242.24) for the optimum tour that starts from vertex 1. This tour has a min AWD value of 14164.07 (non-cyclical min AWD is 13752.16 from vertex 16) if the tour starts from vertex 16. However, the approximate tour from the plain nearest neighbor approximation algorithm that starts from the vertex 43 gives min AWD as 13957.78 (for the rotation that starts from vertex no 17, for the non-cyclical case the min AWD is 13357.82 for the rotation that starts from vertex 17 of the approximate tour that starts from 43). So, in short, the optimum tour neither has the minimum AWD value nor it can give the smallest "min AWD" value as one of its "rotations". Figure 3.2.12 shows an example TSP tour, $\tau = (V_1, \ldots, V_8)$, with 8 vertices and having a cost of $7+3+5+7+4+3+5+4 = 38$. For this example the non-cyclical AWD for $\rho^{F}_{0}(\tau)$ is $\frac{1}{7}(7+10+15+22+26+29+34) = 20.43$. Whereas the cyclical AWD for $\rho^{F}_{0}(\tau)$ is $\frac{1}{8}(7+10+15+22+26+29+34+38) = 22.62$. For this example, the min cyclical AWD(τ) is cyclical AWD$(\rho^{B}_{0}(\tau))$ = cyclical AWD$(V_1, V_8, V_7, V_6, V_5, V_4, V_3, V_2) = 16.12$.

Results

⟨results4⟩

By using different types of datasets and different numbers of vertices, the proposed algorithm is compared to the other standard approximation algorithms, like the nearest neighbor algorithm, various insertion heuristic algorithms, and the 2-Opt algorithm. The algorithms utilized in benchmarks and the abbreviations we used for them in the tables are listed in Table 3.2.4 below. Each of these algorithms is benchmarked 20 times for the same dataset and the average

[3] tsp data: http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsp/index.htmlberlin52.tsp.gz, optimum tour: http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsp/index.htmlberlin52.opt.tour.gz

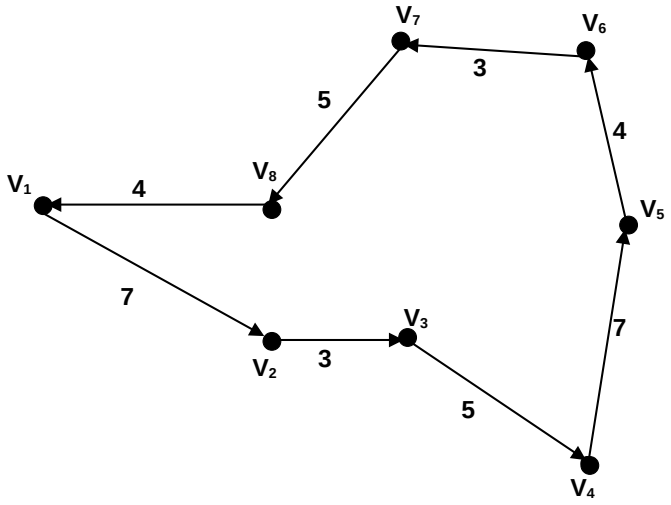


Figure 3.2.12: Example TSP tour on a graph with 8 vertices.

⟨fig:awd-ex⟩

is taken for small datasets (vertices < 1000). For the larger datasets (vertices > 1000) sometimes a single run of the standard algorithm could last several hours (farthest insertion with the rl5915 dataset runs about 9 min and with the usa13509 dataset runs 1.5 hours). This type of benchmarking requires a 24/7 available server with a professional cooling system. For this reason, only the proposed algorithm (with and without heuristic version), arbitrary insertion, and nearest neighbor are benchmarked. According to the small dataset benchmarks, arbitrary insertion and nearest neighbor were the two fastest algorithms we could benchmark. After running these algorithms five times, we took the average of the data we collected. The HW/SW specifications of the system that benchmarks 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 5.12.4
- R version 4.1.0 (2021-05-18) – "Camp Pontanezen"

Two different versions of the proposed algorithm is included in the benchmarks, to see the effect of the proposed 3-edge heuristic. The one with the proposed 3-edge heuristic (CH) and the one without this heuristic (CNH). Both of these versions are deterministic algorithms with zero standard deviation. The other standard algorithms have stochastic elements giving them high standard deviations for the results on the same dataset. The reader is advised to check the detailed results on our GitHub repo (https://github.com/kk-1/tsp) for a better understanding of these claims. The datasets that are utilized can be grouped into three groups: Several of the Symmetric Euclidean datasets from the standard TSPLIB datasets, Tnm* datasets[4] from [68], and the grid and random datasets we created[5]. The custom grid datasets we created are related to our previous study related to drone assisted search and rescue operations [84]. In this study, we proposed a configuration of charging stations in a rectangular and triangular (hexagonal) grid that helps drones to "cover" the desired region on the sea. For this reason, we wanted to see how the proposed algorithm can perform for these datasets. The reader should be aware of the fact that in the case of regular grids no TSP tour is needed to be searched. The tour for

[4] http://www.or.uni-bonn.de/~hougardy/HardTSPInstances.html

[5] Codes, results, and datasets https://github.com/kk-1/tsp

such regular grids can be formulated and the optimum tour cost can be found with the help of the geometric calculations. However, these data sets are valuable for comparison. We generated two different lattice (grid) configurations rectangular (myLattice datasets) and triangular (myHexLattice datasets) lattices and randomly created vertex datasets (myRND datasets). We also wanted to see how the performance can be degraded in case of introducing irregularities to these perfect grids by randomly removing certain percentages of the vertices. As a result, we created irregular grid datasets (myRNDLattice, myRNDHexLattice). The details of the datasets we utilized in the benchmarks are listed in Table 3.2.5.

Table 3.2.4: The list of algorithms utilized in benchmarks.

tab:algo-legend⟩

Symbol	Explanation
CH	The proposed algorithm with 3-edge heuristics (Concave hull with Heuristic)
CNH	The proposed algorithm without 3-edge heuristic (Concave hull No Heuristic)
NI	Nearest Insertion algorithm
FI	Farthest Insertion algorithm
CI	Cheapest Insertion algorithm
AI	Arbitrary Insertion algorithm
NN	Nearest Neighbor algorithm
RNN	Repetitive Nearest Neighbor algorithm
2-Opt	2-Opt algorithm
OPT	The Optimum values known so far

Table 3.2.5: Datasets used in the benchmarks.

⟨tab:dataset-legend⟩

Symbol	Comment[a]	Group	# Vertices	Link to obtain
Small Datasets with number of vertices < 1000				
att48	48 capitals of the US	TSPLIB	48	http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsp/index.html
berlin52	52 locations in Berlin	TSPLIB	52	http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsp/index.html
pr76	76-city problem	TSPLIB	76	http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsp/index.html
kroA100	100-city problem	TSPLIB	100	http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsp/index.html
lin105	105-city problem	TSPLIB	105	http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsp/index.html
ch130	130-city problem	TSPLIB	130	http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsp/index.html
ch150	150-city Problem	TSPLIB	150	http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsp/index.html
a280	Drilling problem	TSPLIB	280	http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsp/index.html
pcb442	Drilling problem	TSPLIB	442	http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsp/index.html
Tnm52	52 geometric points	From [68]	52	http://www.or.uni-bonn.de/~hougardy/HardTSPInstances.html
Tnm76	76 geometric points	From [68]	76	http://www.or.uni-bonn.de/~hougardy/HardTSPInstances.html
Tnm100	100 geometric points	From [68]	100	http://www.or.uni-bonn.de/~hougardy/HardTSPInstances.html
Tnm127	127 geometric points	From [68]	127	http://www.or.uni-bonn.de/~hougardy/HardTSPInstances.html
Tnm154	154 geometric points	From [68]	154	http://www.or.uni-bonn.de/~hougardy/HardTSPInstances.html
Tnm178	178 geometric points	From [68]	178	http://www.or.uni-bonn.de/~hougardy/HardTSPInstances.html
Tnm199	199 geometric points	From [68]	199	http://www.or.uni-bonn.de/~hougardy/HardTSPInstances.html
myRND-100	100 random points	Our custom data	100	https://github.com/kk-1/tsp
myRND-200	200 random points	Our custom data	200	https://github.com/kk-1/tsp
myRND-300	300 random points	Our custom data	300	https://github.com/kk-1/tsp
myRND-400	400 random points	Our custom data	400	https://github.com/kk-1/tsp
myLattice-10x10-100	10x10 Regular grid	Our custom data	100	https://github.com/kk-1/tsp
myLattice-10x20-200	10x20 Regular grid	Our custom data	200	https://github.com/kk-1/tsp
myLattice-15x20-300	15x20 Regular grid	Our custom data	300	https://github.com/kk-1/tsp
myLattice-20x20-400	20x20 Regular grid	Our custom data	400	https://github.com/kk-1/tsp
myRNDLattice-12x12-100	12x12 Irregular grid (31% removal)	Our custom data	100	https://github.com/kk-1/tsp
myRNDLattice-12x23-200	12x23 Irregular grid (28% removal)	Our custom data	200	https://github.com/kk-1/tsp
myRNDLattice-18x23-300	18x23 Irregular grid (28% removal)	Our custom data	300	https://github.com/kk-1/tsp
myRNDLattice-23x23-400	23x23 Irregular grid (24% removal)	Our custom data	400	https://github.com/kk-1/tsp
myHexLattice-10x10-100	10x10 Regular hex grid	Our custom data	100	https://github.com/kk-1/tsp
myHexLattice-10x20-200	10x20 Regular hex grid	Our custom data	200	https://github.com/kk-1/tsp
myHexLattice-15x20-300	15x20 Regular hex grid	Our custom data	300	https://github.com/kk-1/tsp
myHexLattice-20x20-400	20x20 Regular hex grid	Our custom data	400	https://github.com/kk-1/tsp
myRNDHexLattice-12x12-100	12x12 Irregular hex grid (31% removal)	Our custom data	100	https://github.com/kk-1/tsp
myRNDHexLattice-12x23-200	12x23 Irregular hex grid (28% removal)	Our custom data	200	https://github.com/kk-1/tsp
myRNDHexLattice-18x23-300	18x23 Irregular hex grid (28% removal)	Our custom data	300	https://github.com/kk-1/tsp
myRNDHexLattice-23x23-400	23x23 Irregular hex grid (24% removal)	Our custom data	400	https://github.com/kk-1/tsp
Big Datasets with number of vertices > 1000				
pr1002	1002-city problem	TSPLIB	1002	http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsp/index.html
pr2392	2392-city problem	TSPLIB	2392	http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsp/index.html
rl5915	5915-city problem	TSPLIB	5915	http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsp/index.html
usa13509	13509-city problem	TSPLIB	13509	http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsp/index.html
pla33810E	Programmed logic array	Modified TSPLIB	33810	https://github.com/kk-1/tsp
myLattice-100x100-10000	100x100 Regular grid	Our custom data	10000	https://github.com/kk-1/tsp
myLattice-100x200-20000	100x200 Regular grid	Our custom data	20000	https://github.com/kk-1/tsp
myLattice-150x200-30000	150x200 Regular grid	Our custom data	30000	https://github.com/kk-1/tsp
myRNDLattice-105x105-10000	105x105 Irregular grid (9.3% removal)	Our custom data	10000	https://github.com/kk-1/tsp
myRNDLattice-105x210-20000	105x210 Irregular grid (9.3% removal)	Our custom data	20000	https://github.com/kk-1/tsp
myRNDLattice-158x210-30000	158x210 Irregular grid (9.6% removal)	Our custom data	30000	https://github.com/kk-1/tsp
myHexLattice-100x100-10000	100x100 Regular hex grid	Our custom data	10000	https://github.com/kk-1/tsp
myHexLattice-100x200-20000	100x200 Regular hex grid	Our custom data	20000	https://github.com/kk-1/tsp
myHexLattice-150x200-30000	150x200 Regular hex grid	Our custom data	30000	https://github.com/kk-1/tsp
myRNDHexLattice-105x105-10000	105x105 Irregular hex grid (9.3% removal)	Our custom data	10000	https://github.com/kk-1/tsp
myRNDHexLattice-105x210-20000	105x210 Irregular hex grid (9.3% removal)	Our custom data	20000	https://github.com/kk-1/tsp
myRNDHexLattice-158x210-30000	158x210 Irregular hex grid (9.6% removal)	Our custom data	30000	https://github.com/kk-1/tsp

[a] From TSPLIB file whenever it is possible.

Figure 3.2.13 shows an example triangular irregular grid dataset used in the benchmarks. In Figure 3.2.14 an example rectangular irregular grid is shown.

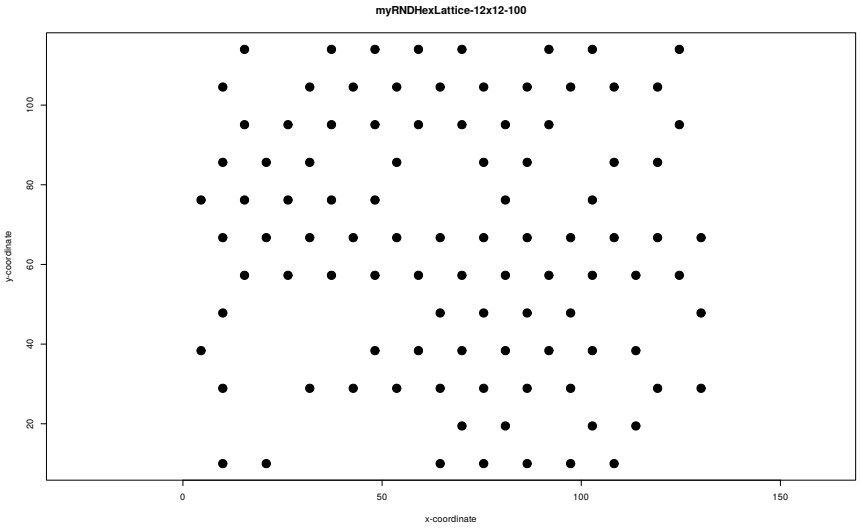


Figure 3.2.13: The custom data set myRNDHexLattice-12x12-100. 12 by 12 (144 vertices) triangular grid in which 44 vertices were randomly selected for removal.

rnd-hex-lattice⟩

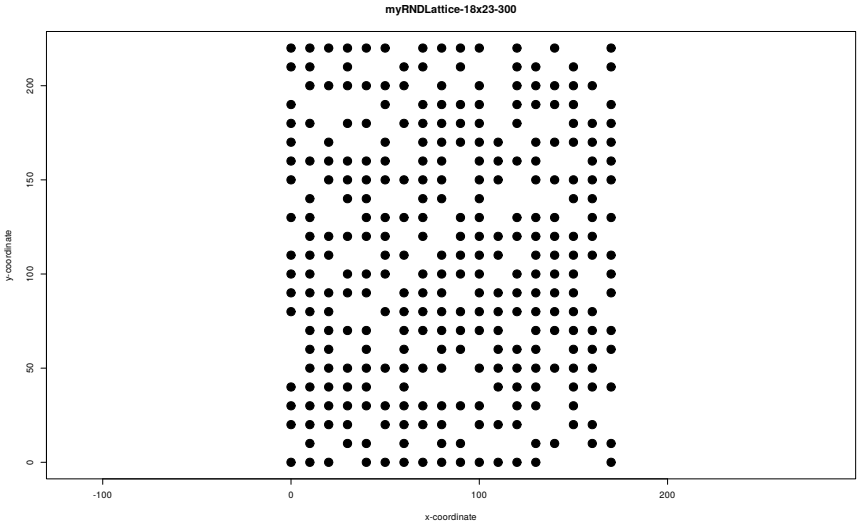


Figure 3.2.14: The custom data set myRNDLattice-18x23-300. 18 by 23 (414 vertices) rectangular grid in which 114 vertices were randomly selected for removal.

fig:rnd-lattice⟩

In the first part of the performance analysis, we wanted to present an example study (not a detailed statistical study) to classify the datasets. Mostly, for the benchmarking of the TSP algorithms, the input consists of varying size (the number of vertices) datasets. However, there can be datasets with different sizes but having the same statistical characteristics. The inclusion of datasets with different characteristics can reveal different performance characteristics of the algorithms. We carried out basic exploratory data analysis on the selected datasets to see if there are some differences and if they can be grouped into different classes. Example methodology for doing such an analysis can be found in [43].

Almost all TSP algorithms need to know edge distances. For the algorithms that the geometric configuration of the vertices matters, the actual coordinates are important. For this reason, in our analysis, the distance values from the distance matrix are selected as the "sample" in our

statistical analysis. As a first step, we investigated the binned histograms of the edge distances. The diagonal distances (self-loops, they are zeros) are ignored for the statistical estimations. We also estimated the mean, median, standard deviation, skewness, and kurtosis of the edge distances. The distributions that the edge distances are coming from are also investigated. The estimated statistics are summarized in Table 3.2.6.

Table 3.2.6: Statistics of the edge distances (from distance matrix) for all datasets. "N": Number of vertices. "med": Median. "std": Standard deviation. "skw": Skewness. "krt": Kurtosis. The distribution that data are fitted listed in the last column.

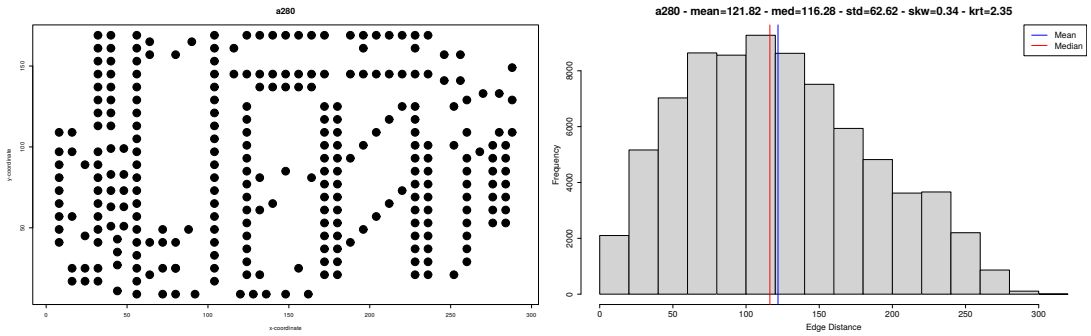
⟨tab:stat-dset⟩

Dataset	N	mean	med	std	skw	krt	Distr
Small Datasets with number of vertices < 1000							
att48	48	3284.64	2943.55	1949.16	0.64	2.61	Weibull
berlin52	52	575.26	525.01	339.91	0.66	2.96	Weibull
pr76	76	7558.71	7107.75	3914.43	0.61	3.17	Weibull
kroA100	100	1710.71	1585.79	916.09	0.43	2.39	Weibull
lin105	105	1177.35	1070.02	670.88	0.48	2.49	Beta
ch130	130	356.22	348.12	169.99	0.24	2.44	Beta
ch150	150	359.31	352.70	169.35	0.17	2.29	Beta
a280	280	121.82	116.28	62.62	0.34	2.35	Beta
pcb442	442	1748.00	1711.72	830.68	0.16	2.32	Beta
Tnm52	52	53416.02	52915.49	25682.51	0.03	1.97	Beta
Tnm76	76	88586.17	89913.18	42928.03	-0.03	1.95	Beta
Tnm100	100	129587.86	131248.04	63044.09	-0.08	1.94	Beta
Tnm127	127	170089.94	173493.08	83136.99	-0.09	1.94	Beta
Tnm154	154	210612.70	216259.12	103273.84	-0.11	1.94	Beta
Tnm178	178	245999.59	250599.25	120884.06	-0.11	1.94	Beta
Tnm199	199	276243.04	282134.72	135950.98	-0.12	1.94	Beta
myRND-100	100	528.56	528.87	249.39	0.08	2.25	Beta
myRND-200	200	1075.32	1051.90	511.02	0.19	2.35	Beta
myRND-300	300	1557.37	1526.43	737.98	0.19	2.36	Beta
myRND-400	400	2083.02	2046.58	989.82	0.19	2.34	Beta
myLattice-10x10-100	100	52.39	50.99	24.26	0.20	2.30	Beta
myLattice-10x20-200	200	80.70	72.80	42.77	0.50	2.55	Beta
myLattice-15x20-300	300	91.89	90.00	44.44	0.28	2.41	Beta
myLattice-20x20-400	400	104.41	101.98	49.31	0.19	2.33	Beta
myRNDLattice-12x12-100	100	63.38	63.25	29.46	0.17	2.29	Beta
myRNDLattice-12x23-200	200	92.52	86.02	48.65	0.51	2.62	Beta
myRNDLattice-18x23-300	300	106.19	102.96	51.01	0.27	2.42	Beta
myRNDLattice-23x23-400	400	121.55	120.42	57.46	0.18	2.33	Beta
myHexLattice-10x10-100	100	54.44	50.87	25.41	0.24	2.34	Normal
myHexLattice-10x20-200	200	80.83	77.70	41.24	0.43	2.51	Normal
myHexLattice-15x20-300	300	90.77	87.86	43.07	0.22	2.35	Normal
myHexLattice-20x20-400	400	102.96	100.64	48.96	0.22	2.36	Normal
myRNDHexLattice-12x12-100	100	60.77	57.77	28.64	0.26	2.38	Normal
myRNDHexLattice-12x23-200	200	93.30	87.33	47.28	0.41	2.52	Normal
myRNDHexLattice-18x23-300	300	105.56	102.33	49.93	0.20	2.33	Normal
myRNDHexLattice-23x23-400	400	117.87	115.26	56.27	0.22	2.36	Normal
Big Datasets with number of vertices > 1000							
pr1002	1002	6435.62	6258.79	3160.87	0.24	2.44	Beta
pr2392	2392	6374.93	6215.50	3125.48	0.25	2.38	Beta
rl5915	5915	7194.63	6911.46	3611.25	0.38	2.62	Beta
usa13509	13509	159409.15	132805.81	109330.27	1.06	3.59	Gamma
pla33810E	33810	278283.57	273026.81	133725.91	0.21	2.39	Beta
myHexLattice-100x100-10000	10000	492.52	482.22	235.70	0.21	2.36	Beta
myHexLattice-100x200-20000	20000	733.57	694.95	378.55	0.42	2.53	Beta
myHexLattice-150x200-30000	30000	849.94	832.19	406.81	0.21	2.36	Beta
myLattice-100x100-10000	10000	521.43	511.57	247.87	0.18	2.34	Beta
myLattice-100x200-20000	20000	804.79	749.47	430.87	0.49	2.56	Weibull
myLattice-150x200-30000	30000	917.31	891.40	447.44	0.27	2.42	Beta
myRNDHexLattice-105x105-10000	10000	517.81	507.66	247.75	0.21	2.36	Beta
myRNDHexLattice-105x210-20000	20000	769.27	728.45	396.95	0.42	2.53	Beta
myRNDHexLattice-158x210-30000	30000	893.47	874.90	427.55	0.21	2.36	Beta
myRNDLattice-105x105-10000	10000	546.92	537.59	260.08	0.19	2.34	Beta
myRNDLattice-105x210-20000	20000	844.77	785.88	452.02	0.49	2.56	Weibull
myRNDLattice-158x210-30000	30000	963.43	936.48	469.47	0.27	2.41	Beta

The big (> 1000 vertices) datasets are placed at the end of the table. The plots of histograms and vertex configuration plots can be downloaded from our GitHub repo[6]. Visual inspection of the histograms for the small datasets (number of vertices < 1000) reveals three distinct groups of distributions profile. In the first group, we can list most of the datasets with positive skewness. For the second group, "att48", "berlin52", and "pr76" datasets have slightly higher positive skewness than the others. By looking at the skewness, we can say that most of the datasets have symmetric distributions, since mostly their skewness is between -1 and -0.5 or between 0.5 and 1. In the third group, Tnm* datasets have negative skewness. One of the big datasets, namely the "usa13509" dataset has the highest positive skewness. Negative skewness indicates a dataset that has relatively few low values, whereas positive skewness indicates a dataset that has relatively few high values. This type of information for the edge distances can be important to the approximation algorithms like Nearest Neighbor (NN). If there are few small edge distances (negative skewness) then such algorithms will probably choose a vertex with high-cost edges connected to others. At least in such datasets, the probability of finding consecutive "cheap" edges is lower.

We also tried to see from which distribution these edge distances are coming from. This is an important analysis for classifying the datasets. The "fitdistrplus" [36], R package is one of the useful tools that make such analysis possible. For this, the first step is a visual inspection of histograms for selecting several candidate distributions. The experienced researcher can select several candidate distributions for further testing by visually inspecting the histogram. Cullen and Frey [30] diagrams can also be helpful for this selection. For this diagram, `descdist()` function of "fitdistrplus" can be used. The other steps that can make the distribution fitting are performing various statistical tests and further plot inspections. The statistical tests like Kolmogorov-Smirnov, Cramer-von Mises, and Anderson-Darling provides "Goodness-of-fit" statistics and Akaike's Information Criterion (AIC) can be used to determine the best-fit distribution. The `gofstat()` function of "fitdistrplus" can provide all these statistics. The distribution that gives minimum values for these statistics is the best fit among the selected candidate distributions. In [36] theoretical and practical details can be found for these methods.

As an example, in Figure 3.2.15 the TSPLIB dataset "a280" vertex configuration (Figure 3.2.15a) and the edge distance histogram (Figure 3.2.15b) is shown.



(a) a280 vertex configuration. **(b)** a280 distance histogram.

⟨sub@f{fggaa2800vtx}⟩? ?⟨sub@f{fggaa2800hist}⟩?

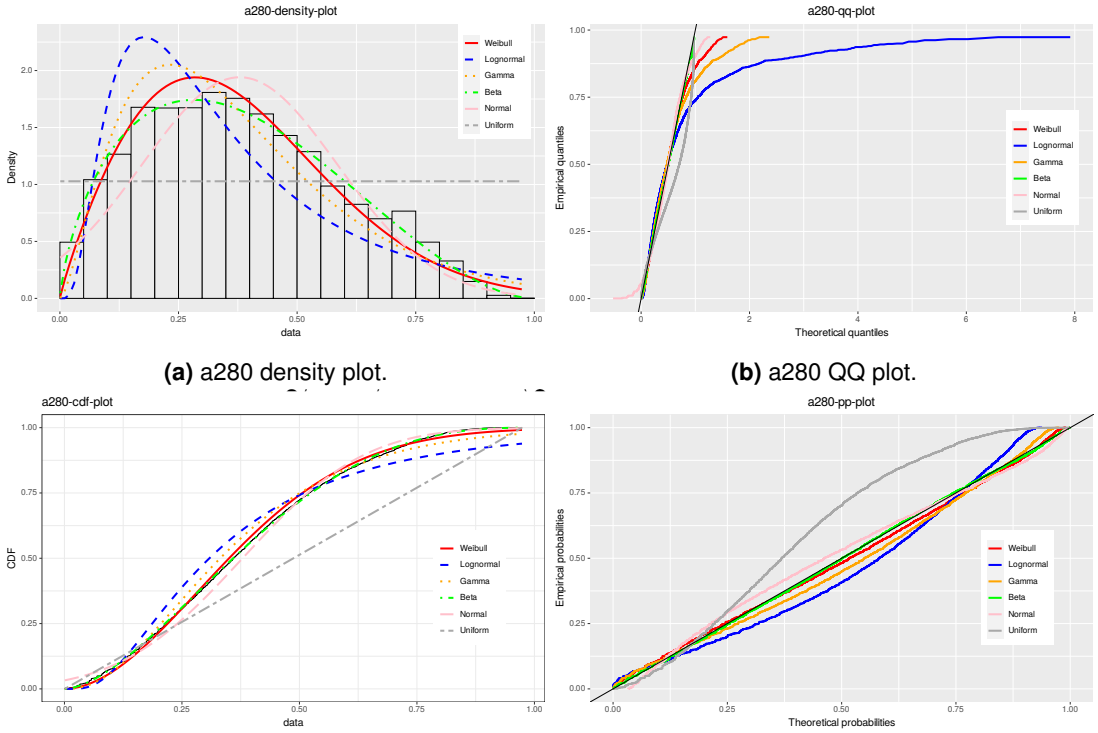
Figure 3.2.15: a280 dataset.

⟨fig:a280⟩

In Figure 3.2.16 important statistical plots that are used in deciding on the best-fit distribution are shown for the TSPLIB dataset "a280". We can see in Figure 3.2.16a that the histogram is very well aligned with the "Beta" distribution. In addition to that, in all other plots, Fig-

[6] https://github.com/kk-1/tsp/raw/master/datasets-hist-vtx-plots-V4.pdf

ure 3.2.16b, 3.2.16c, and 3.2.16d there is a good alignment with the Beta distribution. The test statistics also affirm this observation. We conclude that the edge distances in the a280 dataset are coming from the Beta distribution.



(a) a280 density plot.

(b) a280 QQ plot.

(c) a280 CDF plot.

(d) a280 PP plot.

Figure 3.2.16: a280 statistical plots.

From the distribution column of Table 3.2.6, we can say that there are 3 groups of data if we ignore the "usa13509" TSPLIB dataset. Mostly, we have a "Beta" distribution. Then we have "Normal" and the "Weibull" distributions. The TSPLIB dataset "usa13509" fits the "Gamma" distribution.

In the second part of our performance analysis, we wanted to quantify the effect of the heuristics we proposed. For the datasets we used in the benchmarks, the nearest vertex merging heuristic savings are less frequent compared to the savings from the 3-edge heuristic. To see these savings, we included in the benchmarks the version of the proposed method without a 3-edge heuristic. We also calculated the total cost of the generated rings. The approximate tour cost from the version without a 3-edge heuristic gives us the way to see if there are any savings after the nearest vertex merging heuristic. Then, the approximate tour cost from the proposed method helps us to see the savings that come from the 3-edge heuristic over the nearest vertex merging heuristic. Mostly, the merging introduces cost overhead. However, generally, there is a saving from the 3-edge heuristic. The number of vertices in a specific geometric configuration plays role in these savings. For small (less than 100-vertex) datasets a single ring may be created by the concave hull algorithm. In these cases, the proposed merging heuristics do not provide any savings (only the "remaining" points add to the merging cost).

Table 3.2.7 and 3.2.8 summarizes the savings specific to the datasets we used in the benchmarks. Here the reader should note that the merging process is expected to introduce overhead to the cost of the final merged ring. However, since we saw that there are cases in which the overall cost is reduced after merging, we presented the results as "savings". This shows the

effectiveness of the heuristics we proposed. The explanation of the columns in Table 3.2.7 and 3.2.8 are listed below:

- **CNH Tour Cost**: The cost of the final merged ring with the nearest vertex heuristic only.

- **CH Tour Cost**: The cost of the final merged ring with the addition of the 3-edge heuristic.

- **Merge Savings**: From the "nearest vertex merging" heuristic over the total cost of the rings.

- **Edge Savings**: From the "3-edge" heuristic over the "nearest vertex merging" heuristics.

- **Total Savings**: The total savings over the total cost of the rings.

The nearest vertex merging heuristic keeps the merging cost down, even sometimes saves from the cost. The 3-edge heuristic is proved to be effective in savings. In every dataset, it was able to save from the total cost of the rings. The proposed algorithm, even in small amounts, can save from the total cost of the rings in the overall merging process for small datasets. However, for the bigger datasets, the overall merging process introduced about a 4-10% increase in the cost. But in general, the proposed heuristics in their preliminary versions performed well. Based on the approximate tour costs reported in Table 3.2.11, the proposed algorithm performed second after the farthest insertion algorithm. However, the reader should note that the running time of the proposed algorithm is much better than the standard farthest insertion algorithm when Table 3.2.10 is analyzed. The weak point of the proposed algorithm is in merging the rectangular lattices. A little investigation by tracing the stages of the runs revealed to us that the diagonal connections the proposed heuristic uses in rectangular lattices, over time, add up and introduce high costs in the merging. More elaborated heuristics are needed to overcome this weakness.

Table 3.2.7: Savings from small dataset benchmarks (number of vertices < 1000).

⟨tab:small-save⟩

DataSet	Number of Rings	Total Cost of Rings	CNH Tour Cost	Merge Savings	CH Tour Cost	Edge Savings	Total Savings	CNH-AR	CH-AR	OPT Tour Cost
att48	1	33543.9	35811.2	-6.8%	35618	0.5%	-6.2%	1.068	1.062	33523.7
berlin52	2	9096.7	9013.7	0.9%	9013.7	0.0%	0.9%	1.195	1.195	7544.4
pr76	2	126374.3	118876.7	5.9%	115790	2.6%	8.4%	1.099	1.071	108159.4
kroA100	2	26097.4	23310.1	10.7%	22899.1	1.8%	12.3%	1.095	1.076	21285.44
lin105	2	18704.6	15730.7	15.9%	15596.1	0.9%	16.6%	1.094	1.084	14383
ch130	2	7092.9	7038.8	0.8%	7015	0.3%	1.1%	1.152	1.148	6110.861
ch150	2	7839.2	7309.6	6.8%	7176.5	1.8%	8.5%	1.119	1.099	6532.281
a280	4	3617.4	3448	4.7%	3335.9	3.3%	7.8%	1.333	1.29	2586.77
pcb442	5	57861.6	72425	-25.2%	66961.6	7.5%	-15.7%	1.426	1.319	50783.55
TSPLIB Dataset AVG				1.5%		2.1%	3.7%			
Tnm52	2	577959	629868.7	-9.0%	580277.6	7.9%	-0.4%	1.142	1.052	551609
Tnm76	2	1014327	1031006	-1.6%	982422.9	4.7%	3.1%	1.085	1.034	949961
Tnm100	2	1453919	1518293	-4.4%	1500454	1.2%	-3.2%	1.086	1.073	1398070
Tnm127	2	1933591	2012491	-4.1%	1963559	2.4%	-1.5%	1.076	1.049	1871162
Tnm154	2	2508205	2479303	1.2%	2409694	2.8%	3.9%	1.055	1.025	2350345
Tnm178	2	2977636	3038862	-2.1%	2831484	6.8%	4.9%	1.096	1.021	2771953
Tnm199	2	3371826	3352743	0.6%	3184252	5.0%	5.6%	1.068	1.014	3139778
Tnm* Dataset AVG				-2.8%		4.4%	1.8%			
myRND-100	2	9378.7	7880.5	16.0%	7839.8	0.5%	16.4%	NA	NA	NA
myRND-200	3	26892.6	25424.1	5.5%	24395	4.0%	9.3%	NA	NA	NA
myRND-300	3	48772.5	46552.1	4.6%	44664.1	4.1%	8.4%	NA	NA	NA
myRND-400	4	76384.3	76533.3	-0.2%	72591.6	5.2%	5.0%	NA	NA	NA
myLattice-10x10-100	5	1000	1078.1	-7.8%	1086.4	-0.8%	-8.6%	NA	NA	NA
myLattice-10x20-200	5	2000	2213.0	-10.7%	2086.4	5.7%	-4.3%	NA	NA	NA
myLattice-15x20-300	7	2940	3171.6	-7.9%	3208.8	-1.2%	-9.1%	NA	NA	NA
myLattice-20x20-400	10	4000	4178.1	-4.5%	4215.2	-0.9%	-5.4%	NA	NA	NA
myRNDLattice-12x12-100	3	1223	1363.8	-11.5%	1276.8	6.4%	-4.4%	NA	NA	NA
myRNDLattice-12x23-200	4	2396.2	2834.4	-18.3%	2686.1	5.2%	-12.1%	NA	NA	NA
myRNDLattice-18x23-300	6	3479.1	4276.45	-22.9%	4006.5	6.3%	-15.2%	NA	NA	NA
myRNDLattice-23x23-400	7	4530.7	5964	-31.6%	5465.6	8.4%	-20.6%	NA	NA	NA
myHexLattice-10x10-100	1	1098.9	1121.1	-2.0%	1118.1	0.3%	-1.7%	NA	NA	NA
myHexLattice-10x20-200	1	2205.9	2244.4	-1.7%	2244.4	0.0%	-1.7%	NA	NA	NA
myHexLattice-15x20-300	1	3220	3220	0.0%	3220	0.0%	0.0%	NA	NA	NA
myHexLattice-20x20-400	1	4227.7	4227.7	0.0%	4227.7	0.0%	0.0%	NA	NA	NA
myRNDHexLattice-12x12-100	1	1139.6	1161.4	-1.9%	1158.5	0.2%	-1.7%	NA	NA	NA
myRNDHexLattice-12x23-200	2	2710.4	2424.6	10.5%	2394.9	1.2%	11.6%	NA	NA	NA
myRNDHexLattice-18x23-300	3	3619.5	3808	-5.2%	3632.7	4.6%	-0.4%	NA	NA	NA
myRNDHexLattice-23x23-400	3	4777.8	4735.2	0.9%	4579.1	3.3%	4.2%	NA	NA	NA
Custom Dataset AVG				-4.4%		2.6%	-1.5%			
Overall Small Dataset AVG				-2.6%		2.8%	0.4%			

Table 3.2.8: Savings from big dataset benchmarks (number of vertices > 1000).

⟨tab:big-save⟩

DataSet	Number of Rings	Total Cost of Rings	CNH Tour Cost	Merge Savings	CH Tour Cost	Edge Savings	Total Savings	CNH-AR	CH-AR	OPT Tour Cost
pr1002	4	342149.9	324144.7	5.3%	312125.2	3.7%	8.8%	1.251	1.205	259066.7
pr2392	8	476273.3	533934.1	-12.1%	504587.6	5.5%	-5.9%	1.412	1.335	378062.8
rl5915	14	755515.2	874002.3	-15.7%	812312.9	7.1%	-7.5%	1.545	1.436	565530[a]
usa13509	14	26235348	28380516	-8.2%	26696934	5.9%	-1.8%	1.42	1.336	19982859
pla33810E[b]	32	84395889	104421961	-23.7%	96441315	7.6%	-14.3%	1.581	1.46	66043099
TSPLIB Big Dataset AVG				-10.9%		6.0%	-4.1%			
myLattice-100x100-10000	49	100006.5	101812	-1.8%	102090.1	-0.3%	-2.1%	NA	NA	NA
myLattice-100x200-20000	50	201602.2	207967.3	-3.2%	206916.4	0.5%	-2.6%	NA	NA	NA
myLattice-150x200-30000	74	300082.4	321227.4	-7.0%	346635.5	-7.9%	-15.5%	NA	NA	NA
myRNDLattice-105x105-10000	36	104387.1	162958.4	-56.1%	148750.9	8.7%	-42.5%	NA	NA	NA
myRNDLattice-105x210-20000	38	210060.8	331056.9	-57.6%	300293	9.3%	-43.0%	NA	NA	NA
myRNDLattice-158x210-30000	55	314077.4	495390.9	-57.7%	449158.5	9.3%	-43.0%	NA	NA	NA
myHexLattice-100x100-10000	1	101087.1	101107.3	0.0%	101107.3	0.0%	0.0%	NA	NA	NA
myHexLattice-100x200-20000	2	204587.5	202281.6	1.1%	202274.2	0.0%	1.1%	NA	NA	NA
myHexLattice-150x200-30000	2	308224	302383.4	1.9%	302363.2	0.0%	1.9%	NA	NA	NA
myRNDHexLattice-105x105-10000	4	112901.5	102713.8	9.0%	102372.4	0.3%	9.3%	NA	NA	NA
myRNDHexLattice-105x210-20000	5	223347	205605.8	7.9%	204998.3	0.3%	8.2%	NA	NA	NA
myRNDHexLattice-158x210-30000	4	326373.5	305392.1	6.4%	304855	0.2%	6.6%	NA	NA	NA
Custom Big Dataset AVG				-13.1%		1.7%	-10.1%			

[a] http://www.math.uwaterloo.ca/tsp/rl5915/rl5915_sol.html

[b] The format is modified for our functions:https://github.com/kk-1/tsp

In the third and the final part of our performance analysis, we tabulated benchmark results for the metrics we listed in Section **??** in Tables 3.2.9, 3.2.10, 3.2.11, 3.2.12, 3.2.13,and 3.2.14. The best results are marked with light green colors and the worst results marked with light red color. For Table 3.2.12 we need to use the "highest" and the "lowest" values as we can not theorize that the highest percentage of DT edges signifies the best TSP tour performance. In general, there is no "clear winner" algorithm when these tables are viewed. The algorithm excelling in one metric gave the worst result in other metrics. For small datasets, FI seems to be the best algorithm (the most green in total). However, for the running time, this algorithm is not the best. In fact, for big datasets, it has the worst running time. On the other hand, the proposed algorithm (CH) offered a good overall balanced performance. For big datasets, very long run times prevented us to benchmark all of the algorithms. As we have noted, other algorithms for the big datasets may run hours. For this reason, we run the fastest algorithms 5 times for the big datasets and take the average. We picked the two fastest standard algorithms, AI and NN. Against them, the proposed algorithm performed slightly better than the AI algorithm which was the best among the two standard algorithms we could benchmark. However, these results do not prove the overall superiority of the proposed algorithm. We can only offer the proposed algorithm as a good and very fast approximate solution finder. Still, for the hexagonal lattices, the proposed algorithm performed better than the other algorithms. Although all the tables should be evaluated together, the individual tables are necessary if a certain metric has the most important contribution to the optimization problem. We can only offer the proposed algorithm as a good and very fast approximate solution finder. Although all the tables should be evaluated together, the individual tables are necessary if a certain metric has the most important contribution to the optimization problem.

Table 3.2.9: Approximation ratios against known optimum tour cost. The custom datasets are not included since we do not know the optimum tour cost.

⟨tab:ar⟩

Small Datasets with number of vertices < 1000									
DataSet	CH	CNH	NI	FI	CI	AI	NN	RNN	2-Opt
att48	1.062	1.068	1.131	1.052	1.086	1.066	1.256	1.170	1.117
berlin52	1.195	1.195	1.206	1.084	1.194	1.106	1.228	1.085	1.109
pr76	1.071	1.099	1.202	1.083	1.173	1.073	1.343	1.210	1.099
kroA100	1.076	1.095	1.220	1.075	1.178	1.093	1.272	1.160	1.146
lin105	1.084	1.094	1.271	1.066	1.205	1.099	1.297	1.178	1.135
ch130	1.148	1.152	1.208	1.078	1.172	1.084	1.268	1.178	1.106
ch150	1.099	1.119	1.235	1.094	1.223	1.107	1.184	1.084	1.130
a280	1.290	1.333	1.188	1.142	1.203	1.158	1.303	1.174	1.167
pcb442	1.319	1.426	1.195	1.133	1.172	1.142	1.276	1.181	1.129
Tnm52	1.052	1.142	1.145	1.016	1.032	1.024	1.163	1.108	1.067
Tnm76	1.034	1.085	1.116	1.018	1.067	1.015	1.134	1.091	1.077
Tnm100	1.073	1.086	1.069	1.015	1.093	1.026	1.131	1.076	1.089
Tnm127	1.049	1.076	1.086	1.015	1.080	1.034	1.121	1.088	1.053
Tnm154	1.025	1.055	1.087	1.025	1.067	1.025	1.113	1.076	1.060
Tnm178	1.021	1.096	1.074	1.030	1.067	1.032	1.095	1.057	1.058
Tnm199	1.014	1.068	1.071	1.022	1.062	1.037	1.084	1.052	1.063
# of best	4	0	0	10	0	2	0	1	1
# of worst	0	2	1	0	0	0	13	0	0
Big Datasets with number of vertices > 1000									
DataSet	CH	CNH	NI	FI	CI	AI	NN	RNN	2-Opt
pr1002	1.205	1.251				1.142	1.255		
pr2392	1.335	1.412				1.149	1.273		
rl5915	1.436	1.545				1.229	1.223		
usa13509	1.336	1.420				1.143	1.255		
pla33810E	1.460	1.581				1.177	1.251		
# of best	0	0				4	1		
# of worst	0	5				0	1		

In Table 3.2.9 the approximation ratios of the algorithms are listed. This table contains results from the datasets that the optimum tour cost is known. The FI algorithm performed best and the NN algorithm performed worst overall. The proposed algorithm performed second to the FI algorithm and for the Tnm* datasets it performed well. For the big datasets, the benchmarks are carried out against AI and NN algorithms since they are the fastest algorithms. The proposed algorithms did not offer a better approximation ratio for the datasets in general. However, from Table 3.2.10, it can be seen that for big datasets, the proposed algorithm offers better running time.

Table 3.2.10: Benchmark results for running time in seconds. Contains rounding errors. 0.000s can be regarded as close to 0.001.

⟨tab:runtime⟩

Small Datasets with number of vertices < 1000									
DataSet	CH	CNH	NI	FI	CI	AI	NN	RNN	2-Opt
att48	0.018	0.018	0.003	0.003	0.003	0.000	0.001	0.020	0.000
berlin52	0.024	0.023	0.004	0.004	0.003	0.001	0.000	0.023	0.001
pr76	0.027	0.025	0.007	0.007	0.005	0.000	0.001	0.041	0.001
kroA100	0.029	0.028	0.012	0.012	0.008	0.001	0.001	0.074	0.002
lin105	0.030	0.030	0.013	0.013	0.009	0.001	0.001	0.081	0.002
ch130	0.034	0.033	0.019	0.019	0.014	0.001	0.001	0.131	0.005
ch150	0.037	0.037	0.027	0.027	0.018	0.001	0.001	0.182	0.006
a280	0.070	0.072	0.121	0.121	0.070	0.002	0.003	0.903	0.043
pcb442	0.105	0.104	0.346	0.349	0.200	0.005	0.007	3.193	0.188
Tnm52	0.022	0.054	0.004	0.004	0.003	0.000	0.000	0.023	0.001
Tnm76	0.026	0.058	0.007	0.007	0.005	0.001	0.001	0.042	0.001
Tnm100	0.029	0.061	0.012	0.012	0.008	0.001	0.001	0.073	0.002
Tnm127	0.034	0.032	0.019	0.019	0.013	0.001	0.001	0.129	0.004
Tnm154	0.042	0.035	0.029	0.029	0.019	0.001	0.001	0.193	0.007
Tnm178	0.041	0.040	0.039	0.040	0.026	0.001	0.002	0.279	0.012
Tnm199	0.045	0.043	0.054	0.054	0.033	0.001	0.002	0.360	0.016
myRND-100	0.028	0.061	0.012	0.012	0.008	0.001	0.001	0.074	0.002
myRND-200	0.049	0.080	0.053	0.054	0.031	0.001	0.002	0.357	0.017
myRND-300	0.060	0.091	0.135	0.135	0.081	0.003	0.003	1.075	0.062
myRND-400	0.081	0.112	0.271	0.272	0.158	0.004	0.006	2.200	0.149
myLattice-10x10-100	0.039	0.070	0.012	0.012	0.009	0.001	0.001	0.100	0.002
myLattice-10x20-200	0.058	0.091	0.055	0.054	0.032	0.001	0.003	0.468	0.014
myLattice-15x20-300	0.088	0.116	0.137	0.135	0.081	0.003	0.004	1.265	0.051
myLattice-20x20-400	0.120	0.148	0.276	0.278	0.158	0.004	0.006	2.708	0.123
myRNDLattice-12x12-100	0.033	0.066	0.012	0.012	0.009	0.001	0.001	0.090	0.002
myRNDLattice-12x23-200	0.053	0.084	0.054	0.054	0.032	0.001	0.002	0.437	0.015
myRNDLattice-18x23-300	0.081	0.111	0.135	0.135	0.081	0.004	0.004	1.175	0.053
myRNDLattice-23x23-400	0.104	0.134	0.273	0.274	0.159	0.004	0.006	2.461	0.129
myHexLattice-10x10-100	0.025	0.026	0.012	0.012	0.008	0.001	0.001	0.075	0.002
myHexLattice-10x20-200	0.039	0.038	0.054	0.053	0.032	0.002	0.002	0.362	0.015
myHexLattice-15x20-300	0.046	0.045	0.134	0.137	0.081	0.004	0.004	1.082	0.054
myHexLattice-20x20-400	0.050	0.053	0.273	0.274	0.161	0.004	0.005	2.211	0.133
myRNDHexLattice-12x12-100	0.025	0.025	0.012	0.012	0.008	0.001	0.001	0.078	0.002
myRNDHexLattice-12x23-200	0.044	0.073	0.053	0.055	0.032	0.001	0.002	0.386	0.016
myRNDHexLattice-18x23-300	0.061	0.093	0.133	0.135	0.079	0.004	0.003	1.071	0.056
myRNDHexLattice-23x23-400	0.067	0.098	0.272	0.271	0.160	0.004	0.005	2.213	0.137
# of best	0	0	0	0	0	34	20	0	2
# of worst	2	3	0	0	0	0	0	32	0

Big Datasets with number of vertices > 1000									
DataSet	CH	CNH	NI	FI	CI	AI	NN	RNN	2-Opt
pr1002	0.170	0.163				0.055	0.054		
pr2392	0.463	0.453				0.253	0.214		
rl5915	1.691	1.674				1.302	1.386		
usa13509	6.432	6.407				6.444	6.843		
pla33810E	32.101	33.207				37.417	36.007		
myLattice-100x100-10000	4.855	4.787				3.602	3.461		
myLattice-100x200-20000	14.654	14.428				14.832	14.161		
myLattice-150x200-30000	29.516	28.898				27.577	25.652		
myRNDLattice-105x105-10000	4.344	4.276				3.614	3.495		
myRNDLattice-105x210-20000	13.583	13.480				13.185	12.574		
myRNDLattice-158x210-30000	28.473	27.423				36.921	34.292		
myHexLattice-100x100-10000	0.383	0.383				3.573	3.750		
myHexLattice-100x200-20000	0.684	0.686				12.298	15.043		
myHexLattice-150x200-30000	1.030	1.033				31.938	34.614		
myRNDHexLattice-105x105-10000	0.641	0.633				3.294	3.313		
myRNDHexLattice-105x210-20000	1.563	1.567				11.710	14.389		
myRNDHexLattice-158x210-30000	1.922	1.933				30.321	37.400		
# of best	6	4				1	7		
# of worst	7	0				2	8		

From Table 3.2.11 it can be said that the FI algorithm offers the best tour costs for small datasets. The proposed algorithm offers the best performance for lattice based (especially for hexagonal/triangular lattices) datasets when Tables 3.2.10 and 3.2.11 analyzed together.

Table 3.2.11: Approximation tour costs and the optimum tour cost if it is known.

⟨tab:tourCost⟩

Small Datasets with number of vertices < 1000										
DataSet	CH	CNH	NI	FI	CI	AI	NN	RNN	2-Opt	OPT
att48	35618.0	35811.2	37893.3	35267.5	36391.9	35723.1	42112.7	39236.9	37458.4	33523.7
berlin52	9013.7	9013.7	9097.7	8175.5	9007.3	8346.4	9265.4	8182.2	8368.6	7544.4
pr76	115790.0	118876.7	130028.5	117173.9	126836.5	116098.3	145226.7	130921.0	118838.3	108159.4
kroA100	22899.1	23310.1	25957.0	22874.7	25073.5	23270.7	27084.2	24698.5	24401.9	21285.4
lin105	15596.1	15730.7	18287.9	15331.2	17327.6	15802.6	18646.0	16939.4	16322.2	14383.0
ch130	7015.0	7038.8	7381.6	6588.2	7164.9	6625.8	7747.4	7198.7	6755.5	6110.9
ch150	7176.5	7309.6	8066.9	7148.7	7988.7	7228.8	7734.2	7078.4	7379.4	6532.3
a280	3335.9	3448.0	3072.7	2953.5	3110.7	2995.7	3369.7	3037.7	3018.1	2586.8
pcb442	66961.6	72425.1	60667.9	57537.9	59493.1	58001.5	64819.2	59974.6	57354.4	50783.5
Tnm52	580277.6	629868.7	631392.6	560644.1	569150.6	564563.9	641498.5	611144.5	588427.0	551609.0
Tnm76	982422.9	1031006.4	1059915.3	967492.9	1013572.5	964570.0	1076935.8	1036344.5	1023304.7	949961.0
Tnm100	1500453.9	1518293.0	1494706.9	1419589.7	1528507.9	1434149.2	1581000.3	1504321.2	1522734.7	1398070.0
Tnm127	1963559.0	2012491.4	2032093.6	1899182.6	2021353.1	1935509.7	2098110.4	2035956.1	1970676.4	1871162.0
Tnm154	2409693.6	2479302.9	2554017.2	2410183.8	2507436.4	2408361.3	2615109.8	2530726.9	2491082.3	2350345.0
Tnm178	2831483.6	3038862.3	2977078.3	2853729.5	2959051.6	2859530.9	3034734.3	2929236.0	2932518.6	2771953.0
Tnm199	3184252.0	3352742.7	3361460.2	3208718.4	3333595.6	3255338.8	3401967.8	3303246.1	3336879.5	3139778.0
myRND-100	7839.7	7880.5	9141.5	7734.6	8566.5	7943.3	8629.9	7937.3	8274.3	NA
myRND-200	24395.0	25424.1	26466.8	23309.5	24952.4	23762.4	26540.6	24663.4	24283.1	NA
myRND-300	44664.1	46552.1	48398.5	42569.2	46160.9	43501.2	49364.4	46904.6	43946.9	NA
myRND-400	72591.6	76533.3	76418.7	66707.4	72391.4	67688.1	74849.1	72025.3	68329.4	NA
myLattice-10x10-100	1086.4	1078.1	1086.2	1054.7	1061.7	1053.0	1229.8	1116.6	1098.0	NA
myLattice-10x20-200	2086.4	2213.0	2178.5	2116.8	2131.4	2116.5	2488.1	2275.8	2217.4	NA
myLattice-15x20-300	3208.8	3171.6	3255.7	3195.1	3189.7	3188.6	3671.5	3438.4	3259.6	NA
myLattice-20x20-400	4215.2	4178.1	4353.8	4231.3	4257.6	4255.7	4900.1	4597.3	4356.5	NA
myRNDLattice-12x12-100	1276.8	1363.8	1230.0	1139.6	1220.9	1166.2	1361.0	1227.5	1220.8	NA
myRNDLattice-12x23-200	2686.1	2834.4	2462.9	2319.7	2428.5	2337.6	2724.5	2495.3	2405.6	NA
myRNDLattice-18x23-300	4006.5	4276.5	3692.0	3479.7	3680.6	3514.7	4062.3	3745.8	3604.7	NA
myRNDLattice-23x23-400	5465.6	5964.0	4798.5	4543.7	4737.0	4569.1	5271.7	4924.4	4697.8	NA
myHexLattice-10x10-100	1118.1	1121.1	1194.6	1114.1	1115.7	1126.3	1276.4	1217.9	1176.3	NA
myHexLattice-10x20-200	2244.4	2244.4	2399.1	2236.4	2230.2	2254.5	2499.0	2335.3	2327.3	NA
myHexLattice-15x20-300	3220.0	3220.0	3279.6	3244.0	3220.0	3265.8	4039.5	3811.5	3362.4	NA
myHexLattice-20x20-400	4227.7	4227.7	4284.0	4244.9	4241.6	4271.8	4548.7	4377.5	4351.2	NA
myRNDHexLattice-12x12-100	1158.5	1161.4	1279.8	1169.2	1229.8	1197.4	1361.5	1253.4	1236.8	NA
myRNDHexLattice-12x23-200	2394.9	2424.6	2577.0	2364.9	2413.0	2391.0	2802.0	2576.2	2443.4	NA
myRNDHexLattice-18x23-300	3632.7	3808.0	3610.6	3413.9	3561.8	3472.2	3972.9	3666.1	3509.6	NA
myRNDHexLattice-23x23-400	4579.1	4735.2	4743.1	4472.5	4611.0	4499.8	5187.4	4862.5	4571.3	NA
# of best	7	4	0	21	2	3	0	1	1	NA
# of worst	0	7	2	0	0	0	27	0	0	NA
Big Datasets with number of vertices > 1000										
DataSet	CH	CNH	NI	FI	CI	AI	NN	RNN	2-Opt	OPT
pr1002	312125.2	324144.7				295787.6	325144.0			259066.0
pr2392	504587.6	533934.1				434582.6	481307.6			378062.0
rl5915	812312.9	874002.2				695270.9	691453.0			565530.0
usa13509	26696933.9	28380516.0				22830164.5	25073156.2			19982859.0
pla33810E	96441314.7	104421961.3				77739880.6	82659720.5			66043099.0
myLattice-100x100-10000	102090.1	101812.0				106450.5	120717.8			NA
myLattice-100x200-20000	206916.4	207967.3				212875.4	240164.6			NA
myLattice-150x200-30000	346635.5	321227.4				319586.4	359707.1			NA
myRNDLattice-105x105-10000	148750.9	162958.4				108860.2	122525.4			NA
myRNDLattice-105x210-20000	300293.0	331056.9				217796.4	245246.0			NA
myRNDLattice-158x210-30000	449158.5	495390.9				326860.4	367166.7			NA
myHexLattice-100x100-10000	101107.3	101107.3				101978.8	103929.1			NA
myHexLattice-100x200-20000	202274.2	202281.6				203932.2	205700.8			NA
myHexLattice-150x200-30000	302363.2	302383.4				304805.6	311072.5			NA
myRNDHexLattice-105x105-10000	102372.0	102713.8				103663.0	115571.8			NA
myRNDHexLattice-105x210-20000	204998.3	205605.8				207111.5	228938.4			NA
myRNDHexLattice-158x210-30000	304855.0	305392.1				309985.6	335015.4			NA
# of best	7	2				8	1			NA
# of worst	0	7				0	10			NA

Table 3.2.12 lists the percentage of the DT edges in the final tours. We can see that not all the optimum tours have 100% DT edges. Also, the highest DT percentage does not always give the best tour cost. For example, the custom big dataset "myLattice-100x200-20000" has the best tour cost found as 206916.4 (Table 3.2.11) by the proposed algorithm with 99.42% DT edges (Table 3.2.12). However, this is not the highest DT edge percentage. The highest percentage is given in Table 3.2.12 as 99.53% from the proposed algorithm without the 3-edge heuristic.

Table 3.2.12: Percentage of tour edges that come from Delaunay Triangulation edges.

⟨tab:delEdgeP⟩

Small Datasets with number of vertices < 1000										
DataSet	CH	CNH	NI	FI	CI	AI	NN	RNN	2-Opt	OPT
att48	95.83	95.83	90.21	98.85	93.96	96.46	88.12	89.58	96.56	100.00
berlin52	88.46	88.46	90.96	97.69	91.44	94.23	91.44	98.08	96.64	100.00
pr76	90.79	88.16	83.68	93.36	89.41	92.11	89.54	89.47	94.14	97.37
kroA100	92.00	89.00	92.35	96.65	91.20	95.10	90.35	94.00	96.85	99.00
lin105	96.19	95.24	86.81	97.05	87.48	94.19	89.38	90.48	95.43	98.09
ch130	90.77	89.23	85.54	96.46	88.15	94.50	92.19	92.31	95.69	98.46
ch150	94.67	93.33	90.43	96.63	90.23	94.83	91.70	93.33	96.93	100.00
a280	81.43	80.36	80.86	81.21	78.46	80.05	81.88	82.09	81.84	88.93
pcb442	91.63	90.95	90.86	95.34	92.92	93.81	91.73	92.48	95.87	99.32
Tnm52	94.23	92.31	89.81	95.39	91.64	94.71	88.27	90.39	94.81	NA
Tnm76	97.37	93.42	91.97	94.80	95.33	94.74	94.41	95.72	95.07	NA
Tnm100	96.00	95.00	93.00	95.45	96.50	95.45	94.70	95.05	94.85	NA
Tnm127	98.42	96.85	91.46	95.47	94.25	95.47	96.81	97.60	95.67	NA
Tnm154	97.40	97.40	92.50	96.43	95.65	95.84	96.92	97.60	96.10	NA
Tnm178	97.19	96.63	91.01	96.43	95.17	96.07	97.75	98.68	96.43	NA
Tnm199	97.49	96.98	91.66	96.56	95.98	96.08	98.17	98.92	96.08	NA
myRND-100	88.00	84.00	77.55	89.45	82.50	86.75	84.35	85.60	88.15	NA
myRND-200	89.50	88.50	87.70	96.60	88.47	94.33	90.50	91.00	96.67	NA
myRND-300	90.67	88.33	88.02	96.27	90.23	93.68	90.58	91.17	96.08	NA
myRND-400	89.50	86.50	87.90	96.36	89.45	94.25	93.08	94.65	96.96	NA
myLattice-10x10-100	95.00	95.00	90.85	93.00	92.15	93.45	89.20	91.75	91.25	NA
myLattice-10x20-200	98.00	96.50	90.05	92.53	92.50	92.80	89.83	91.95	92.80	NA
myLattice-15x20-300	96.67	97.67	90.65	91.82	91.92	92.42	90.13	91.97	92.42	NA
myLattice-20x20-400	97.50	97.50	90.28	92.89	91.67	92.47	90.42	91.71	91.92	NA
myRNDLattice-12x12-100	88.00	88.00	91.25	96.15	92.70	94.50	89.95	92.45	94.50	NA
myRNDLattice-12x23-200	87.50	88.00	90.60	95.42	94.30	95.10	90.97	91.90	94.92	NA
myRNDLattice-18x23-300	87.00	90.00	91.17	95.93	93.72	95.20	90.47	91.70	95.00	NA
myRNDLattice-23x23-400	84.00	90.00	91.54	95.71	93.54	95.11	90.71	92.06	94.86	NA
myHexLattice-10x10-100	99.00	99.00	91.15	99.65	99.35	98.45	96.45	97.90	97.50	NA
myHexLattice-10x20-200	99.00	99.00	90.35	99.42	99.42	98.47	97.75	98.95	98.08	NA
myHexLattice-15x20-300	100.00	100.00	97.68	99.63	100.00	98.83	89.48	89.55	98.78	NA
myHexLattice-20x20-400	100.00	100.00	98.36	99.40	99.33	98.75	98.70	99.14	98.83	NA
myRNDHexLattice-12x12-100	98.00	98.00	90.05	98.65	95.65	97.50	93.05	95.80	97.20	NA
myRNDHexLattice-12x23-200	94.50	95.00	87.20	97.60	94.88	95.97	91.28	92.95	96.58	NA
myRNDHexLattice-18x23-300	92.33	90.67	93.12	97.67	94.38	96.42	93.45	95.50	97.47	NA
myRNDHexLattice-23x23-400	94.00	92.00	91.69	97.50	93.41	96.33	93.28	94.11	97.55	NA
# of best	8	6	0	12	2	0	0	5	9	NA
# of worst	5	5	20	0	2	0	6	0	0	NA
Big Datasets with number of vertices > 1000										
DataSet	CH	CNH	NI	FI	CI	AI	NN	RNN	2-Opt	OPT
pr1002	89.32	89.72				93.81	92.36			99.40
pr2392	86.75	85.95				94.79	93.12			99.54
rl5915	91.80	90.87				93.68	96.08			NA
usa13509	84.53	85.46				94.19	92.10			NA
pla33810E	85.50	89.17				93.92	93.22			99.82
myLattice-100x100-10000	99.47	99.48				92.28	90.92			NA
myLattice-100x200-20000	99.42	99.53				92.40	90.90			NA
myLattice-150x200-30000	99.41	99.63				92.28	90.94			NA
myRNDLattice-105x105-10000	82.47	93.24				93.54	90.89			NA
myRNDLattice-105x210-20000	82.02	93.49				93.40	90.90			NA
myRNDLattice-158x210-30000	80.73	93.13				93.35	90.97			NA
myHexLattice-100x100-10000	100.00	100.00				98.94	99.90			NA
myHexLattice-100x200-20000	100.00	100.00				98.94	99.91			NA
myHexLattice-150x200-30000	99.98	99.98				98.97	99.28			NA
myRNDHexLattice-105x105-10000	98.74	98.69				97.87	95.80			NA
myRNDHexLattice-105x210-20000	98.59	98.47				97.95	95.75			NA
myRNDHexLattice-158x210-30000	99.24	99.09				97.93	97.73			NA
# of best	6	7				6	1			NA
# of worst	6	2				3	6			NA

Table 3.2.13 presents results for the AWD costs of the approximate/optimum tours from

vertex number 1. Again, the standard algorithm FI offered the best values for AWD. However, the proposed algorithm also offered similar performance with less running time (Table 3.2.10) for many datasets. Especially, for the big hexagonal/triangular lattices offered the best values for the AWD metric.

Table 3.2.13: AWD (from vertex 1) costs for the approximations and for the optimum tour cost if it is known.

⟨tab:awd1⟩

Small Datasets with number of vertices < 1000										
DataSet	CH	CNH	NI	FI	CI	AI	NN	RNN	2-Opt	OPT
att48	21416.0	21470.2	20206.1	19139.0	19479.9	17202.8	22812.8	22960.9	18490.8	15623.1
berlin52	4111.3	4111.3	4678.1	4092.1	4614.7	4278.4	4880.8	3961.1	4234.1	4131.8
pr76	58352.5	60288.6	65189.4	59469.2	63884.4	59065.9	81847.0	76904.9	60986.3	54312.6
kroA100	11254.5	11374.5	13095.1	11508.0	12712.3	11750.1	14880.1	14461.1	12207.5	11094.0
lin105	8514.0	8603.3	9261.4	7720.4	8933.9	8005.4	9809.8	10073.3	8357.3	5898.9
ch130	3590.0	3571.1	3708.9	3340.6	3596.3	3342.6	3695.5	3462.8	3425.0	3074.2
ch150	3574.9	3598.7	4056.8	3607.2	4018.9	3635.2	3959.3	3814.3	3734.1	3414.8
a280	1637.0	1692.3	1547.3	1474.2	1554.3	1500.2	1718.2	1547.8	1514.6	1296.1
pcb442	33057.1	33900.1	30234.4	28956.1	29991.4	28171.1	31600.0	30499.3	28498.4	24555.6
Tnm52	278626.4	281077.3	324362.8	285713.7	293259.8	287874.0	326406.3	318939.8	300860.0	NA
Tnm76	521971.5	550384.2	521804.8	494967.7	495400.0	489642.9	493276.2	503979.6	527860.4	NA
Tnm100	732289.9	713079.8	760415.4	717416.7	797623.5	729425.4	755433.0	755153.2	749387.1	NA
Tnm127	935408.0	928280.9	1051298.8	950150.5	1056584.8	973106.2	1009580.4	1064427.4	989283.6	NA
Tnm154	1476570.4	1519319.9	1275011.3	1213902.5	1207890.3	1203017.0	1321794.2	1321087.3	1282495.3	NA
Tnm178	1457755.7	1554836.0	1528233.2	1432038.3	1578013.7	1367097.8	1477065.9	1541955.6	1444823.1	NA
Tnm199	1950600.8	2064895.5	1701019.3	1570300.5	1792088.2	1614463.2	1862625.4	1548837.9	1691681.2	NA
myRND-100	4186.1	4181.0	4610.8	3951.9	4320.5	4044.9	4478.1	4079.6	4093.9	NA
myRND-200	12081.5	13009.2	13290.3	11676.0	12481.5	11957.6	13093.4	11754.6	12293.7	NA
myRND-300	21422.7	21928.3	24282.6	21307.6	23138.2	21822.4	24413.3	24149.7	22208.3	NA
myRND-400	36670.7	38394.4	38462.0	33412.2	36392.7	33926.6	39597.8	37673.4	34416.3	NA
myLattice-10x10-100	542.4	534.2	548.9	532.5	536.0	532.1	617.1	562.0	556.9	NA
myLattice-10x20-200	1038.7	1115.2	1095.5	1063.5	1071.6	1064.6	1262.6	1141.5	1116.6	NA
myLattice-15x20-300	1626.4	1593.8	1635.8	1602.3	1598.2	1596.8	1835.3	1748.6	1635.1	NA
myLattice-20x20-400	2127.9	2095.6	2179.9	2119.1	2133.9	2129.6	2463.5	2325.2	2171.3	NA
myRNDLattice-12x12-100	636.7	677.9	626.5	577.2	618.6	585.8	690.7	625.6	615.9	NA
myRNDLattice-12x23-200	1375.2	1366.8	1232.9	1162.9	1222.7	1174.2	1398.2	1272.7	1210.7	NA
myRNDLattice-18x23-300	2002.7	2108.6	1856.2	1742.3	1842.9	1767.2	2067.8	1924.2	1808.7	NA
myRNDLattice-23x23-400	2697.3	2917.4	2403.7	2274.2	2373.7	2286.1	2671.8	2469.6	2365.8	NA
myHexLattice-10x10-100	566.5	568.4	601.8	562.5	563.4	568.9	654.5	611.6	590.9	NA
myHexLattice-10x20-200	1128.6	1128.5	1204.5	1124.8	1120.3	1132.7	1298.7	1210.2	1166.2	NA
myHexLattice-15x20-300	1615.4	1615.4	1643.6	1629.4	1615.4	1638.1	1997.5	1893.1	1701.7	NA
myHexLattice-20x20-400	2121.5	2121.5	2145.7	2126.9	2125.6	2140.2	2300.2	2223.7	2191.6	NA
myRNDHexLattice-12x12-100	573.6	574.2	646.5	592.1	618.3	604.0	682.8	616.8	622.6	NA
myRNDHexLattice-12x23-200	1208.2	1222.7	1294.8	1190.5	1219.7	1198.7	1405.3	1321.2	1231.6	NA
myRNDHexLattice-18x23-300	1841.4	1918.1	1811.7	1710.9	1788.7	1739.0	1982.7	1863.5	1765.5	NA
myRNDHexLattice-23x23-400	2277.1	2330.7	2378.6	2241.5	2305.7	2257.8	2626.7	2472.3	2268.7	NA
# of best	8	6	0	15	2	6	0	2	0	NA
# of worst	0	6	4	0	2	0	21	3	0	NA
Big Datasets with number of vertices > 1000										
DataSet	CH	CNH	NI	FI	CI	AI	NN	RNN	2-Opt	OPT
pr1002	155801.5	161421.5				147469.2	154162.8			127442.5
pr2392	254129.4	268243.0				216448.4	242943.2			191180.5
rl5915	400243.0	426382.8				350030.1	356452.3			NA
usa13509	14939518.7	15860224.7				10513988.7	11681633.6			NA
pla33810E	48043819.3	51882395.0				38785087.8	41142626.5			32737492.6
myLattice-100x100-10000	51480.3	51229.1				53252.2	61279.9			NA
myLattice-100x200-20000	105215.3	106214.3				106454.1	117257.1			NA
myLattice-150x200-30000	174575.9	162882.1				159800.9	178280.2			NA
myRNDLattice-105x105-10000	73385.6	79729.5				54428.3	61570.8			NA
myRNDLattice-105x210-20000	149310.3	162782.8				108963.1	119323.0			NA
myRNDLattice-158x210-30000	223808.1	244788.7				163426.4	183916.6			NA
myHexLattice-100x100-10000	50562.3	50562.3				50997.9	52703.6			NA
myHexLattice-100x200-20000	101147.5	101148.7				101965.0	103353.1			NA
myHexLattice-150x200-30000	151201.1	151211.3				152390.0	156827.3			NA
myRNDHexLattice-105x105-10000	51222.1	51408.9				51844.0	57182.3			NA
myRNDHexLattice-105x210-20000	102592.8	102911.4				103582.5	114852.4			NA
myRNDHexLattice-158x210-30000	152462.1	152764.7				155027.1	170602.3			NA
# of best	7	2				9	0			NA
# of worst	0	8				0	9			NA

Table 3.2.14 lists the measured "minAWD" values of the approximate/optimum tours. It is no surprise that the standard RNN algorithm offers the best values for minAWD since it iteratively tries to find the "next minimum edges" for the approximate tour construction. In this way, the algorithm kind of construct a chain of the cheapest edges in the approximate tour, giving a tour with a good potential of having a better AWD value. After the standard FI algorithm, the proposed algorithm also offered approximate tours with good AWD potentials. The RNN algorithm different than the NN checks the "next minimum edge" starting from every vertex instead of choosing the initial vertex randomly. The NN algorithm, following the "next minimum edge" heuristic, mostly ends up increasing the global cost of the tour when it tries to

return to the starting vertex.

Table 3.2.14: minAWD costs for the approximations and for the optimum tour cost if it is known.

⟨tab:minAwd⟩

DataSet	CH	CNH	NI	FI	CI	AI	NN	RNN	2-Opt	OPT
att48	14944.0	14966.5	16187.0	14879.0	15473.1	14881.0	15656.1	15394.3	14834.0	14164.1
berlin52	3370.2	3370.2	3867.8	3051.5	3781.7	3277.2	3109.4	3390.5	3213.7	2886.3
pr76	50702.9	52310.1	56653.1	50959.0	56060.1	51611.1	47269.8	46157.4	52205.9	47619.1
kroA100	10601.3	10784.7	11989.0	10746.4	11597.7	10762.8	11131.1	10196.9	10996.4	9888.3
lin105	6898.1	6948.3	7951.9	6372.4	7476.2	6825.1	7229.1	6811.9	7026.3	5898.9
ch130	3354.0	3363.3	3520.8	3090.2	3414.9	3146.6	3087.4	3002.3	3138.8	2895.1
ch150	3431.5	3461.2	3832.8	3405.7	3814.2	3442.6	3355.3	3201.0	3458.8	3100.6
a280	1606.2	1650.0	1452.4	1427.0	1484.8	1446.5	1427.0	1390.2	1414.8	1268.9
pcb442	31239.9	33242.1	28668.4	27535.3	27942.7	27545.3	26776.1	26693.8	27270.5	24555.6
Tnm52	268311.1	273981.4	289735.7	269112.6	271378.8	270438.0	272734.8	276879.9	268843.2	NA
Tnm76	419353.3	442277.4	452310.3	445632.6	412862.7	440434.2	410697.0	400453.8	452550.4	NA
Tnm100	611014.7	596123.7	642231.1	626321.4	558334.2	626560.3	562242.7	545573.3	651392.4	NA
Tnm127	770548.4	783841.9	842076.4	821323.7	719660.2	841154.7	709113.6	692298.0	822994.5	NA
Tnm154	899561.3	922353.0	1042394.3	1031106.1	861716.7	1015757.3	863836.7	839927.7	1023499.1	NA
Tnm178	1022775.2	1035250.3	1184921.1	1229678.5	1015779.4	1188748.3	985603.5	965966.1	1202194.2	NA
Tnm199	1135822.1	1152231.9	1338844.9	1369683.7	1115558.8	1362766.1	1098109.7	1077988.6	1346742.2	NA
myRND-100	3583.3	3597.2	4283.0	3583.6	4006.4	3675.0	3739.9	3657.7	3718.9	NA
myRND-200	11681.5	12222.8	12758.3	11205.8	11820.3	11350.7	11405.8	11283.5	11419.7	NA
myRND-300	21066.7	21808.4	23331.2	20490.5	22321.9	20857.0	21230.4	20889.2	20703.8	NA
myRND-400	35074.2	36724.8	37238.0	32228.6	34964.1	32657.5	31864.0	31139.8	32455.3	NA
myLattice-10x10-100	531.6	526.5	535.8	525.2	526.1	523.3	543.5	531.4	531.6	NA
myLattice-10x20-200	1035.2	1047.9	1075.0	1052.4	1055.9	1050.8	1096.3	1067.2	1062.5	NA
myLattice-15x20-300	1557.0	1547.8	1610.2	1587.9	1582.7	1583.1	1631.7	1606.2	1590.7	NA
myLattice-20x20-400	2067.2	2059.1	2156.5	2101.9	2111.2	2114.8	2175.7	2146.2	2124.7	NA
myRNDLattice-12x12-100	611.0	656.0	592.0	557.9	591.3	570.0	582.0	568.1	579.9	NA
myRNDLattice-12x23-200	1229.6	1322.8	1207.6	1139.6	1194.2	1147.4	1171.4	1149.0	1157.2	NA
myRNDLattice-18x23-300	1960.5	1994.8	1808.9	1711.9	1813.0	1729.6	1730.6	1722.3	1736.1	NA
myRNDLattice-23x23-400	2607.2	2790.2	2364.8	2244.7	2336.8	2254.2	2285.5	2269.6	2280.3	NA
myHexLattice-10x10-100	560.6	560.7	590.1	560.7	560.9	562.9	577.9	569.9	565.5	NA
myHexLattice-10x20-200	1122.8	1122.9	1185.3	1117.9	1116.7	1123.4	1156.6	1125.8	1125.5	NA
myHexLattice-15x20-300	1615.4	1615.4	1633.0	1620.2	1615.4	1627.1	1756.2	1764.0	1633.6	NA
myHexLattice-20x20-400	2115.3	2115.3	2130.3	2121.2	2120.3	2129.0	2162.6	2145.6	2130.7	NA
myRNDHexLattice-12x12-100	573.0	573.5	617.6	575.6	590.3	585.5	610.5	600.5	586.5	NA
myRNDHexLattice-12x23-200	1177.8	1183.0	1264.4	1163.5	1185.4	1176.6	1252.4	1220.6	1184.6	NA
myRNDHexLattice-18x23-300	1749.7	1789.3	1777.1	1683.9	1758.0	1710.6	1750.3	1731.2	1701.9	NA
myRNDHexLattice-23x23-400	2253.4	2293.9	2334.5	2208.4	2266.8	2221.5	2316.0	2281.3	2222.5	NA
# of best	7	4	0	11	2	1	0	13	1	NA
# of worst	0	7	20	2	0	0	5	1	1	NA
Big Datasets with number of vertices > 1000										
DataSet	CH	CNH	NI	FI	CI	AI	NN	RNN	2-Opt	OPT
pr1002	150799.1	156263.7				141384.3	140147.0			123587.2
pr2392	246205.9	260499.6				212723.8	212734.1			186341.2
rl5915	388418.5	414967.3				332834.1	304160.7			NA
usa13509	11193325.8	11911309.3				9558629.8	10556166.3			NA
pla33810E	46982540.9	50564818.2				37992477.9	36124855.1			31974553.6
myLattice-100x100-10000	50558.0	50525.1				53118.8	54794.6			NA
myLattice-100x200-20000	101539.5	101529.0				106306.3	110603.2			NA
myLattice-150x200-30000	158928.8	155947.0				159619.7	164711.4			NA
myRNDLattice-105x105-10000	72824.0	79046.9				54302.6	55794.4			NA
myRNDLattice-105x210-20000	148006.3	161720.6				108687.7	111671.4			NA
myRNDLattice-158x210-30000	222948.8	244525.6				163193.2	167435.6			NA
myHexLattice-100x100-10000	50555.0	50555.0				50936.4	50791.8			NA
myHexLattice-100x200-20000	101129.6	101133.0				101894.6	101537.6			NA
myHexLattice-150x200-30000	151160.6	151170.3				152293.1	152939.5			NA
myRNDHexLattice-105x105-10000	51109.6	51263.9				51757.4	53906.6			NA
myRNDHexLattice-105x210-20000	102343.5	102601.8				103421.0	107014.6			NA
myRNDHexLattice-158x210-30000	152318.7	152568.7				154831.5	158443.6			NA
# of best	6	4				5	3			NA
# of worst	0	6				3	7			NA

3.2.5 Conclusions

⟨conc4⟩

We proposed a novel deterministic heuristic algorithm for the TSP which is based on the iterative and vertex-exhaustive concave hull construction. In the first phase, the proposed algorithm constructs concentric concave hulls (rings) of the "cities" (vertices) iteratively and merges them by using novel heuristic selection-merging methods. The iterative construction of the rings provides a useful way to sort and buffer vertices according to geometric distances that speeds up the selection and the merging phase. In the "merging" phase, we utilized two novel heuristics for selecting vertices and for on-the-fly path optimization. These novel heuristics can be classified as "nearest neighbor vertex merging" and "3-edge" on-the-fly minimum path selection, which is a form of the standard 3-opt heuristic. We analyzed the complexity of the proposed algorithm. We presented an assessment of the savings that can be obtained from the proposed

heuristics. The benchmarks are carried out against the other standard approximation algorithms (nearest insertion, 2-Opt, etc...) by considering various metrics we presented in Section **??** by using standard TSPLIB datasets and custom datasets. Among the metrics that are used, to our knowledge, our article is the first to consider the AWD of the "cities", which are the second party in TSP optimization. We have shown with examples that the optimum TSP tour does not always have the minimum AWD. We also investigated "min AWD", a novel metric that assesses the potentiality of the tour concerning the AWD. In Section **??** we provided details on these metrics.

The proposed algorithm offers a good compromise of the metrics we considered in the benchmarks against other standard approximation algorithms. It offered an approximation ratio less than 1.5 for the datasets having vertices from 50 up to 30000. Approximation ratio of 1.5 is considered the worst-case limit for the polynomial TSP algorithms, which is studied by Christofides in [24]. Especially, it showed an exceptional performance in the hexagonal/triangular lattice type datasets.

As future work, we can propose studying the performance of the proposed algorithm with various "concavity" parameters. The theoretical proof for the approximation ratio of the proposed algorithm is also a valuable research practice. Novel "graph-density" metrics can be associated with the data sets for additional benchmarking studies for various levels of density.

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

- It proposes a new and flexible geometry-based paradigm that can be integrated and extended with various existing and new heuristics for the TSP approximation algorithms.
- It reflects on the multi-party and multi-objective optimization scheme for the TSP and offers various novel metrics to be considered.
- It proposes analyzing various statistical/geometrical properties of the data sets other than their sizes for better benchmarking.

4. Conclusions

⟨chap:conc⟩

Two roads diverged in a wood, and I—
I took the one less traveled by,
And that has made all the difference.

"The Road Not Taken" by Robert Frost

The thesis offered quantitative research by following the Positivist paradigm. The course of research work was structured by the classical "Question, Hypothesis, Experiment, Conclusions, and Report" phases of the Scientific Method. Proposed methods were benchmarked in an experimental setting. Proofs were presented related to several methods. Case studies were used as presentation tools for the proposed methods.

The content of the thesis centered on the geometry-based optimization heuristics for two fundamental drone-based operations. Namely, region coverage and pathfinding. The research work of the thesis proposed a flexible Evolutionary Algorithm (EA) based multi-party multi-objective optimization scheme for single/multi-BS disaster region communication coverage and a framework for optimized heuristic drone pathfinding over an optimized charging station grid for "visiting" entities in a "covered region".

Region Coverage is a special case of the classical SCP ($\mathcal{NP}$-Hard). It is a geometric covering case with extra constraints (i.e. min overlap) for optimization. We proposed multi-party multi-objective priority-based heuristic optimization based on EAs that can approximate solutions quickly to overcome complex formulation (for optimization) and "$\mathcal{NP}$-Hardness". Three different EAs are benchmarked in the optimization framework. Namely DEoptim [9], GA [129], and GenSA [143]. The summary of the main results is given in Table 4.0.1.

Table 4.0.1: Results summary for the region coverage research.

ov-general-results⟩

EA	Search Type	Best results in
DEoptim	Population	Run time
GA	Population	Sum of drone dist. (energy)
GenSA	Single solution	Coverage

For the rescue pathfinding, we proposed optimum (min number and "no blind spot") CS grid configurations (Tri/Sq) to cover the operation region. The CS grid not only increases the operational range of the drone but also creates synergy with the proposed heuristic pathfinding algorithm, redGraySP, which "saves path length" in the range of 10-17% over the "base case". The proposed novel concave hull-based TSP heuristic algorithm [86], concaveTSP, achieved an approximation ratio of less than 1.5 for the datasets varying from 50 up to 30000 vertices. The summary of the main results is given in Table 4.0.2. In short, Tri Grid gives better "relative"

(marked with * in Table 4.0.2) savings and better coverage, but the “rescue tour” is more expensive.

Table 4.0.2: Results summary for the rescue pathfinding research. Best results are marked with green.

general-results⟩

Type	Metric	Tri Grid	Sq Grid
Theoretical SingleBoat	**Prob. of having a Good RG Path**	0.78	0.82
	Prob. of using a Good RG Path	0.45	0.31
	Savings 1-way	43%	64%
	Savings return	20%	50%
	Area per CS	1.30 $unit^{2\dagger}$	0.5 $unit^2$
Simulations MultiBoat	**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)

† 1 unit = Drone range

For the rest of the chapter in Section 4.1, contributions and impacts of the thesis are discussed. Future works are listed in Section 4.2 for the interested reader.

4.1 Impacts of the Thesis

⟨final-contrib⟩ The context of case studies utilized in the thesis not only provided experimental verification of the abstractions related to the proposed methods but also provided novel contributions to the real-life problems. In this manner, the objectivity of the models and the usefulness of the thesis are both strengthened. The general contributions and impacts of the thesis can be summarized in a shortlist as follows:

- **Addressed** limited on-board energy issues for EVs.
- **Contributed** with various novel geometry-based optimization heuristics to the effective operations with drones.
- Will have **practical impacts** on Sustainability, Smart Cities, IoT, Industry 4.0, effective use of EVs.
- Will have **theoretical impacts** on SCP and Geometry-based optimizations.

Summaries of specific contributions and impacts are given in Section 4.1.1 for the Region Coverage case study and in Section 4.1.2 for the Pathfinding case study.

4.1.1 Impacts of the Region Coverage Case Study

⟨imp-rg⟩ The region coverage case study is presented in the context of providing urgent and/or temporary communication coverage to the mission region by using a fleet of drones (Chapter 2). The main contributions of this study can be listed as follows:

- Novel and essential objectives for the coverage: **Overflow, overlap, sum of drone distances (energy).**
- Performance improvement: **Circle Packing** algorithm for the initial solutions to EAs.
- **Scenario-based weighted scoring** for the fitness function.
- **Multi BS** coverage framework for **Voronoi Tessellated region**, utilization of homogeneous/heterogeneous BSs.

The region coverage case study offers a generic solution to the SCP which is one of the $\mathcal{NP}$-Hard problems. In the classical SCP, the complete coverage with the minimum number of subsets is sought. The solution offered in the region coverage study involved a time-critical geometry-based approximate solution. The coverages of the individual drones are modeled as projected circles on the mission region as a function of their altitudes. The EA-based optimization framework offers approximate solutions to "cover" the polygonal region with these "circles" based on the multi-objective scheme. In the proposed optimization framework, solutions are searched according to the supplied number of drones for the mission. However, the framework also proposes the minimum necessary number of drones (at the "medium" altitude) for complete (100%) coverage. The SCP arises in many application areas [25]. The list of important application areas of the SCP can be given as follows:

- Base Station deployment for cellular networks (i.e. GSM networks) and wireless networks (i.e. 802.11 networks) [138, p. 3].
- Energy efficient monitoring in Wireless Sensor Networks [38].
- Scheduling [92].

- Optimum industrial cutting [26].
- VLSI testing [71].

In this sense, the region coverage work not only will have direct contributions and impacts on the urgent disaster/temporary situation communication coverage, but also the application areas of the SCP will benefit from this research. The EA-based framework provides the flexibility of searching for "shallow solutions" (fewer iterations) and "deep solutions" (many iterations). For the time-critical operations, shallow solutions can be used and for better approximations, deep solutions can be searched with the proposed framework. The communication coverage is not the only application for Region Coverage operations. A contaminated region can be "covered" with drones for identification of the problems and assessing the risks associated with the contamination [53]. In such cases, the response team can operate with greater safety. In various Precision Agriculture applications, drones are necessary to "cover" and monitor the fields [63].

4.1.2 Impacts of the Pathfinding Case Study

⟨imp-rescue⟩ The pathfinding case study (Chapter 3) is presented in the context of rescuing boats on the sea in case of emergencies by sending drones that can provide necessary material and communication. This study proposed an optimized CS Grid and heuristic pathfinding for the optimized rescue tour of the calling boats. We proposed two different optimized CS Grid types. Namely, Tri and Sq grids. The optimization objective for the CS Grid configuration was the complete coverage of the mission region with the minimum number of CSs (homogeneous). Probabilistic analysis is provided for the comparisons between Tri and Sq grids. Results from simulations were also compiled and presented in tables. The pathfinding heuristic is a combination of the proposed TSP heuristic (concaveTSP) which finds the optimum rescue order of the boats and the proposed red-gray edge heuristic (redGraySP) which finds the shortest path from one boat to another jumping from one CS to another. The case study presented the "single BS and single drone" variation of the problem, which is a TSP case at the core. The variation with multiple drones involves VRP (Vehicular Routing Problem) theory. The proposed TSP heuristic is a novel approximation heuristic that benefits from the geometry of the boats. It constructs concentric concave hulls and heuristically merges them into a single tour. The red-gray edge heuristic is designed as a flexible add-on for any pathfinding algorithm. The main contributions of the rescue pathfinding research can be given as list as follows:

- Proposal of an **optimal (min CS - no blind spot) static CS deployment** that can be used for any EV region coverage context.
- Proposal of a novel metric called **Coverage Effectiveness**: metric to assess the effectiveness of regular CS Grid configurations.
- Proposal of the **novel custom TSP (concaveTSP) and pathfinding heuristics (redGraySP)**.
- Proposal of a novel metric called **AWD**: Speed independent metric for assessing the "customer satisfaction" side of the operation.
- Proposal of the **flexible multi-party multi-objective optimization framework for heuristic pathfinding over optimal CS Grid**.

The general impact areas of the rescue pathfinding research can be given as:

- Optimized coverage framework for general EFV: Cost savings and less carbon footprint.

- Bigger operation range/area for EFVs.
- Optimized delivery operations with EFVs.
- Optimized inspection operations with EFVs.
- Optimized Precision Agriculture with EFVs.
- Optimized Disaster Management with EFVs.
- Theoretical impacts in: SCP, TSP, and pathfinding (routing).

The proposed framework for drone-based rescue operations is a flexible framework that can also be applied to EVs on the ground. Figure 4.1.1 presents current (as of 2022) statistics about electric car ranges. If for a healthy battery the capacity is kept at 50% this makes the average range of a generic electric car about 160km. So for effective and healthy utilization of electric cars CS, centers should be spaced according to this distance. The proposed optimized CS Grid deployment strategies for drone-based rescue operations can be adapted to electric cars.

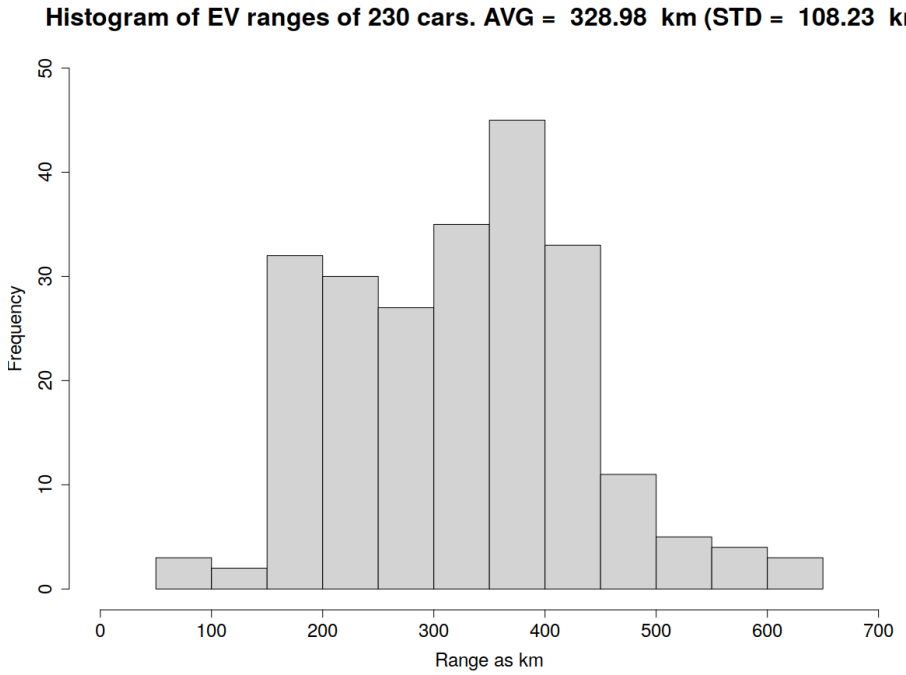


Figure 4.1.1: Histogram of 230 electric cars from the online database at https://ev-database.org/.

⟨fig:ev-stat⟩

The proposed TSP heuristic algorithm [86], concaveTSP, is benchmarked against the other standard approximation algorithms (nearest insertion, 2-opt, etc...) according to metrics we presented in our research by using standard TSPLIB datasets and custom datasets we generated. The benchmark datasets are selected to cover different sizes, different geometries, and different statistical characteristics. The metrics are selected by considering a general multi-party multi-objective optimization problem. As part of the performance assessment, we proposed a novel metric AWD that is related to the "cities". We have shown that the optimum TSP tour does not always have the minimum AWD. We also investigated another novel metric that assess the potentiality of the tour concerning the AWD, which we called minAWD.

It is the minimum AWD of all the possible "rotations" (discussed in Section 3.2) in both directions (clockwise/anti-clockwise) of a tour. The proposed TSP heuristic algorithm gave an approximation ratio of less than 1.5 for the datasets varying from 50 up to 30000 vertices. In addition to that performance characteristic, it showed exceptional performance in the lattice type datasets. The novelties and contributions of the research related to the concaveTSP can be listed as follows:

- It proposes a new and flexible geometry-based paradigm that can be integrated with various existing and new heuristics for the TSP approximation algorithms.
- It reflects on the multi-party and multi-objective optimization scheme for the TSP and offers various novel metrics to be considered.
- It proposes analyzing various statistical/geometrical properties of the data sets other than their sizes for better benchmarking.

The TSP is also one of the $\mathcal{NP}$-Hard problems that can be reduced from the Hamiltonian Circuit which was among the 21 $\mathcal{NP}$-Complete problems listed by Karp in [79]. The problem has many application areas in operations research, management science, electronics and robotics industry, data analysis in psychology, and, X-Ray crystallography [75, 62, 102]. One of the application areas of TSP related to drone-based operations is the so-called "drone parcel delivery" or "last-mile parcel delivery" [4, 125]. In these type of operations, the drones deliver parcels directly from the depot. Another alternative is the delivery from the truck loaded with parcels going to strategic delivery points. The delivery logistics involves optimization of the route of the delivery drone over many delivery points. The proposed concaveTSP algorithm can be utilized for these types of operations, generating real-time approximate TSP routes for the delivery plan. The route of the drone can be quickly updated even in the case of a new delivery request while the drone is flying. The other drone-based operations that will be impacted by the proposed TSP heuristic algorithm can be listed as:

- **Optimized delivery with drones [136, 23]:** Delivery of parcels with drones is a cost-saving and flexible operation. Drones can avoid high ground traffic in the cities and can reach places where ground vehicles can not reach. Optimized path for many delivery points saves much time and energy, especially for daily deliveries.
- **Optimized inspection with drones [76, 1, 115]:** Monitoring and maintenance of Wind Turbines, Cell Towers, and Industrial and Construction Sites need frequent inspections. Drones provide less cost, greater security, and access to areas that are difficult to access with traditional methods. With the proposed concaveTSP algorithm, the "inspection tour" that requires many entities can be optimized for greater savings
- **Precision Agriculture [31]:** Drones are used to monitor agricultural fields without harming the crops (as they fly over them) and with less cost. Such operations that require monitoring many sites daily can benefit from the proposed concaveTSP algorithm.
- **Disaster Management [50]:** In disaster management operations, the drone deployment system should be able to plan the path for drones quickly and accurately as possible. The strategic points for collecting sensor data should be visited in an optimized manner for a quick response to disasters.

In this sense, the proposed concaveTSP method will have an impact on many application areas of the TSP.

The novel pathfinding heuristic, redGraySP, utilizes the red and gray dynamic constraint edges for path savings. This heuristic is an example of how the optimized CS Grid can be used in synergy with the pathfinding method for energy savings. It is a flexible add-on for any path searching algorithm, like Dijkstra's Shortest Path and A*. This novelty will have a theoretical impact on the topic of Fuel Constrained UAV Routing Problem (FCURP) [135]. Practical impacts will be on the general use of UAVs. The proposed pathfinding heuristic can provide a more effective use of UAVs. Especially for routine delivery, inspection, and monitoring operations, the proposed heuristics can make the use of drones more effective.

4.2 Future Works

⟨final-fworks⟩ The research done for the thesis proposed flexible optimization frameworks for drone-based operations. In many ways, the proposed frameworks can be modified and extended for various applications. However, we can give a list of extensions and modifications for which interested readers can work for novel contributions.

- Region Coverage:

  1. For initial solutions homogeneous drone altitudes (homogeneous projected coverage circles) were considered. Heterogeneous drone altitudes can be tried for generating initial solutions. The effect of utilizing such initial solutions on the approximate solution generation can be researched.
  2. The multi BS coverage framework considered homogeneous BSs. Consideration of heterogeneous BSs can be tried by utilizing "Weighted Voronoi Tessellation".

- Pathfinding:

  1. The CS Grid configure by considering homogeneous CSs. Heterogeneous CS Grid configuration can be studied and the proposed heuristics can be modified for this type of CS Grid configuration.
  2. Currently, the redGraySP heuristic is designed as an add-on for any generic shortest path algorithm. In this sense, there can be some trade-offs. For future work, we can suggest integrating it fully and creating a special shortest path algorithm. This algorithm should work with dynamic data structures.
  3. Dijkstra's Shortest Path algorithm does not consider the best "direction" and uses a "flooding" type search. Heuristics like A* considers the best direction and avoids "flooding". In this sense, better pathfinding can be achieved if the proposed heuristics can be integrated into A* type search algorithms.
  4. The "jumps" from one boat to another are not considered in the proposed framework. For this, augmenting the graph data structures with "yellow" edges to represent possible jumps among boats that are very close to each other can be considered. 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-optimized path.

5. The proposed framework, currently, considers a single BS and single drone case. 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 [88] for dividing the large mission region into smaller regions based on the BS positions. By using this division scheme the drone-based operations can be done in parallel over each sub-region.

Appendices

A. Poster

⟨fig:poster⟩ The poster on the next page is prepared for the 1st Computer Science and Mathematics Workshop - CSMW 2022 (https://computerscience.unicam.it/csmw2022). The original size was A0 size paper.

Geometry-Based Optimization Heuristics for Region Coverage and Pathfinding in Drone-Based Operations

Kemal İhsan Kılıç and Leonardo Mostarda (Supervisor)
{kemal.kemal, leonardo.mostarda}@unicam.it
Computer Science Division, University of Camerino

Methods and Results

Region Coverage

Region Coverage is a special "Set Cover" ($\mathcal{NP}$-Hard) problem. We proposed multi-party multi-objective priority-based heuristic optimization based on EAs that can approximates solution quickly to overcome complex formulation (for optimization) and "$\mathcal{NP}$-Hardness" [1,2].

Region coverage: Simulations Summary

EA	Soln Type	Best for
DEoptim	Population	Run time
GA	Population	Σ of drone dist. (energy)
GenSA	Single soln	Coverage

Boat Rescue: CS Grid and Pathfinding

We proposed optimum (min number and "no blind spot") CS grid configurations (Tri/Sq) to cover the operation region. The CS grid not only increases the operational range of the drone but also creates synergy with the proposed heuristic pathfinding algorithm, redGraySP, which "saves path length" in the range of 10-17% over the "base case". Proposed novel concave hull based TSP heuristic algorithm, concaveTSP, achieved approximation ratio of less than 1.5 for the datasets varying from 50 up to 30000 vertices [3,4,5].

Pathfinding: Simulations Summary

Type	Metric	Tri Grid	Sq Grid
Theoretical Single-Boat	Prob. of having a Good RG Path	0.78	0.82
	Prob. of using a Good RG Path	0.45	0.31
	Savings 1-way	43%	64%
	Savings return	20%	30%
	Area per CS	130 unit^2*	0.5 unit^2
Sim./Statistics Multi-Boat	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)

* 1 unit = Drone range

Tri gives better "relative" savings and better coverage, but the "tour" is more expensive.

Contributions

Region Coverage

- Novel and essential objectives for the coverage: **Overflow, overlap, sum of drone distances (energy).**
- Performance improvement: **Circle Packing** algorithm for the initial solutions to EAs.
- **Scenario-based weighted scoring** for the fitness function.
- **Multi BS** coverage framework for **Voronoi Tessellated region**, utilization of homogeneous/heterogeneous BSs.

Boat Rescue: CS Grid and Pathfinding

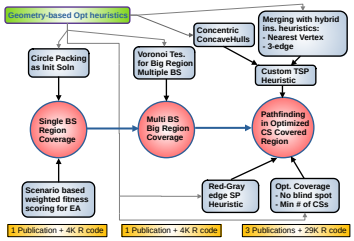
- **Static optimal (min CS - no blind spot) CS Grid** adjusted to drone range for complete region coverage.
- Novel **Coverage Effectiveness** metric for CS Grid.
- **Custom TSP (concaveTSP) + redGraySP pathfinding heuristics**.

Research Summary

Drones are versatile and can be used as mobile IoT platforms. However, the limited onboard energy requires optimization to increase operational effectiveness.

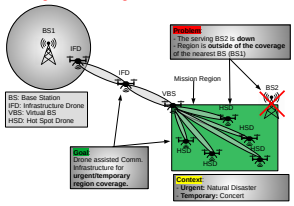
Static Operation: Region coverage is vital for communication in natural disasters or temporal events.

Dynamic Operation: Rescue operations require "region coverage" with CSs for the "reachability" of the drones. The "optimum" rescue path should be found quickly.

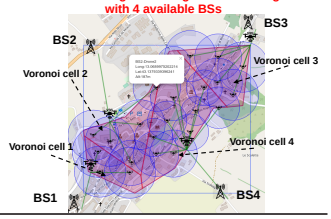


We proposed a flexible Evolutionary Algorithm (EA) based multi-party multi-objective optimization scheme for **single/multi-BS disaster region communication coverage** (first row) and a framework for **optimized heuristic drone pathfinding over an optimized charging station grid** for "visiting" entities in a "covered region" (second and bottom row).

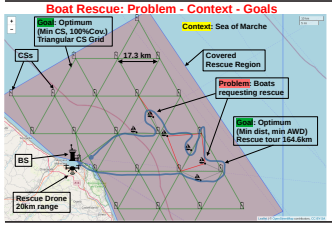
SingleBS Coverage: Problem - Context - Goals



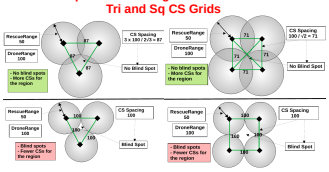
MultiBS Coverage: Coverage of divided region with 4 available BSs



Boat Rescue: Problem - Context - Goals

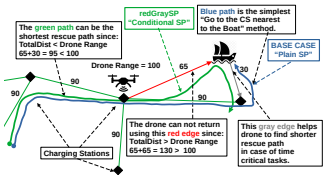


Optimum coverage conditions for Tri and Sq CS Grids

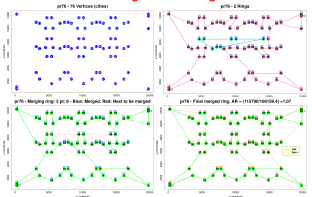


Min number CS - no blind spot region coverage

Heuristic idea used for redGraySP



ConcaveTSP: Stages for finding TSP tour



References

[1] K. I. Kilic, O. Gemikonakli, and L. Mostarda. Multi-objective Priority Based Heuristic Optimization for Region Coverage with UAVs. In AINA, Advances in Intelligent Systems and Computing, 1151:768–779. Springer, 2020.

[2] 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. Journal of Web and Grid Services, 17(2):152–178, 2021.

[3] K. I. Kilic and L. Mostarda. K. I. Kilic and L. Mostarda. Optimum Path Finding Framework for Drone Assisted Boat Rescue Missions. In AINA, Lecture Notes in Networks and Systems, 227:219–231. Springer, 2021.

[4] K. I. Kilic and L. Mostarda. Heuristic Drone Pathfinding Over Optimized Charging Station Grid. IEEE Access, 9:164070–164089, 2021.

[5] K. I. Kilic and L. Mostarda. Novel Concave Hull-Based Heuristic Algorithm For TSP. Operations Research Forum, Springer Nature, 3(2):25, 2022.

B. List of Publications and Code Availability

⟨ch:lop⟩ The list of articles that are published on the thesis and the available research material (code and data) associated with them are given below:

1. K. I. Kilic, O. Gemikonakli, and L. Mostarda. Multi-objective Priority Based Heuristic Optimization for Region Coverage with UAVs. In L. Barolli, F. Amato, F. Moscato, T. Enokido, and M. Takizawa, editors, Advanced Information Networking and Applications, Advances in Intelligent Systems and Computing, 1151:768–779. Springer International Publishing, 2020.
   DOI: https://doi.org/10.1007/978-3-030-44041-1_68 [87]

   Simulator that is written in R language and the results for the single-BS region coverage project can be obtained from: https://github.com/kk-1/drone-coverage

2. K. I. Kilic, O. Gemikonakli, and L. Mostarda. Voronoi Tesselation-based load-balanced multi-objective priority-based heuristic optimization for multi-cell region coverage with UAVs. International Journal of Web and Grid Services, 17(2):152–178, 2021.
   DOI: https://doi.org/10.1504/IJWGS.2021.114574 [88]

   Simulator that is written in R language and the results for the multi-BS region coverage project can be obtained from: https://github.com/kk-1/drone-coverage

3. K. I. Kilic and L. Mostarda. Optimum Path Finding Framework for Drone Assisted Boat Rescue Missions. In L. Barolli, I. Woungang, and T. Enokido, editors, Advanced Information Networking and Applications, Lecture Notes in Networks and Systems, 227:219–231. Springer International Publishing, 2021.
   DOI: https://doi.org/10.1007/978-3-030-75078-7_23 [84]

   Simulator that is written in R language and the results for the boat rescue project can be obtained from: https://github.com/kk-1/boat-rescue

4. K. I. Kilic and L. Mostarda. Heuristic Drone Pathfinding Over Optimized Charging Station Grid. IEEE Access, 9:164070–164089, 2021.
   DOI: https://doi.org/10.1109/ACCESS.2021.3134459 [85]

   Simulator that is written in R language and the results for the boat rescue project can be obtained from: https://github.com/kk-1/boat-rescue

5. K. I. Kilic and L. Mostarda. Novel Concave Hull-Based Heuristic Algorithm For TSP. Springer Nature Operations Research Forum, 3(2):25, 2022.
   DOI: https://doi.org/10.1007/s43069-022-00137-9 [86]

   TSP algorithm that is implemented in R language and the results for the TSP project can be obtained from: https://github.com/kk-1/tsp

C. Simulator Screenshots

?⟨fig:gui⟩? The screenshots of the simulators that are developed for the thesis research are shown in Figures C.0.1 to C.0.3 below.

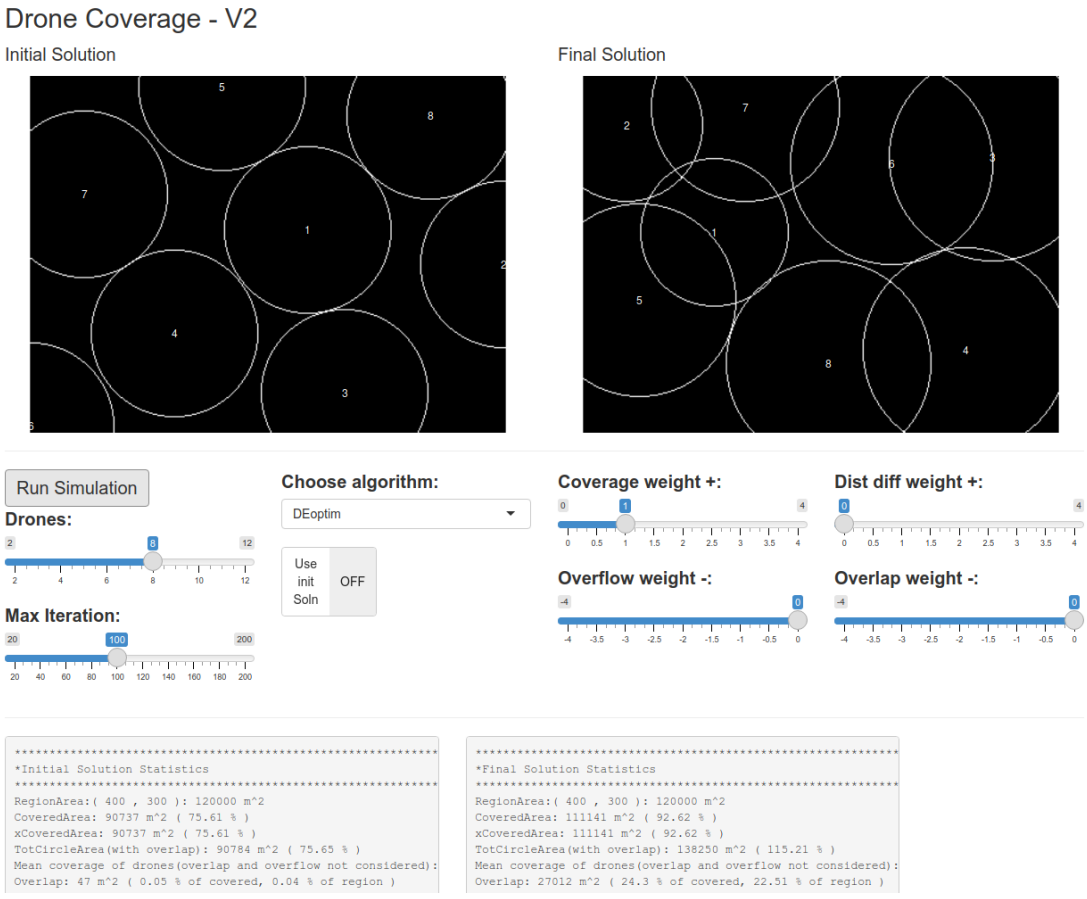


Figure C.0.1: GUI for the single-BS simulator.
Code is available at: https://github.com/kk-1/drone-coverage.

⟨fig:singleBS-gui⟩?

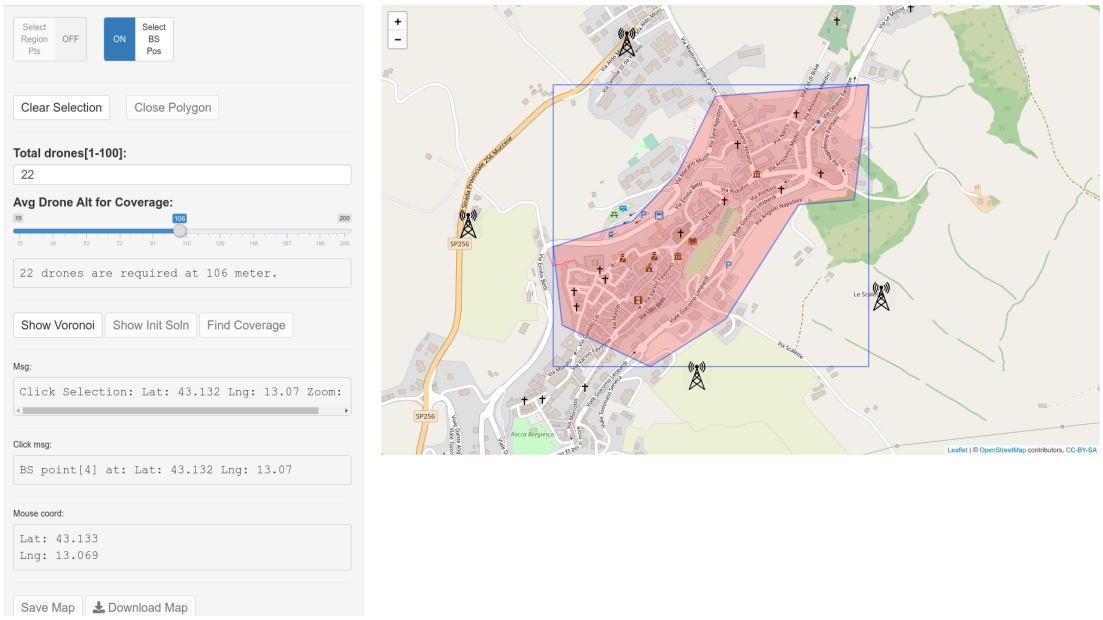


Figure C.0.2: GUI for the multi-BS simulator.
Code is available at: https://github.com/kk-1/drone-coverage.

ig:multiBS-gui⟩?

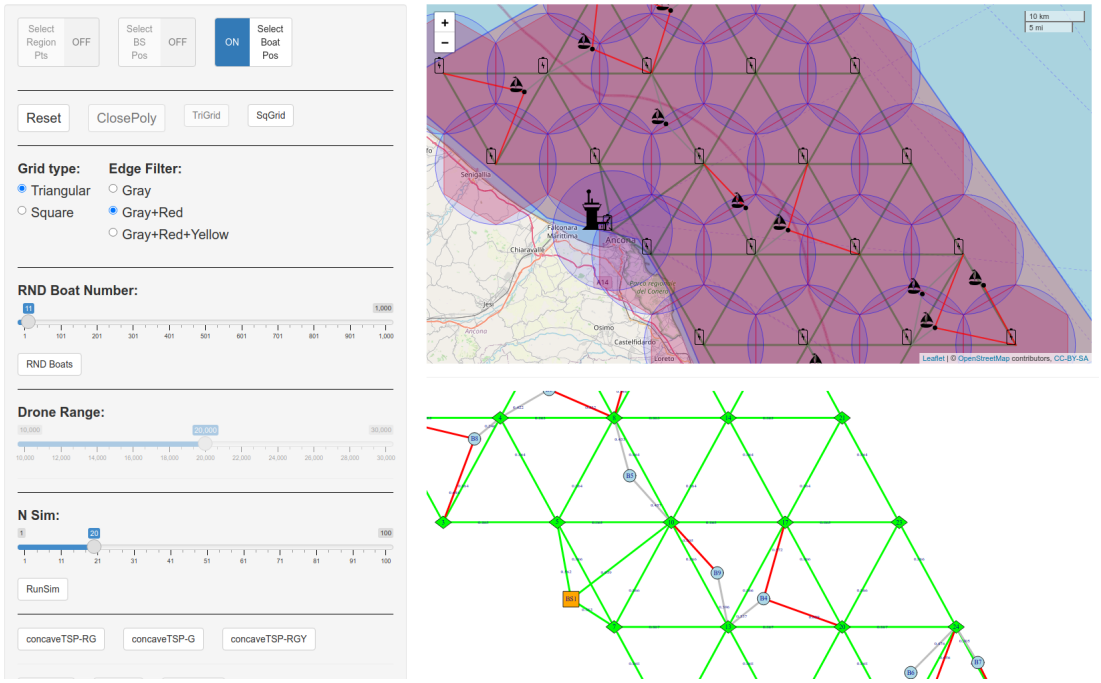


Figure C.0.3: GUI for the boat rescue simulator.
Code is available at: https://github.com/kk-1/boat-rescue.

fig:rescue-gui⟩?

Bibliography

celltower [1] abjdrones. Drone Cell Tower Inspection, Survey, Thermal Imaging and LIDAR. abjdrones news, 2022. https://abjdrones.com/drone-cell-tower-inspection-services/, online; accessed: 14-April-2022.

adamatzky2012 [2] A. Adamatzky. Slime mould computes planar shapes. *International Journal of Bio-Inspired Computation*, 4(3):149–155, 2012. doi:https://doi.org/10.1504/IJBIC.2012.047239.

agarwal2019 [3] P. K. Agarwal, E. Ezra, and K. Fox. Geometric Optimization Revisited. In B. Steffen and G. Woeginger, editors, *Computing and Software Science: State of the Art and Perspectives*, pages 66–84. Springer International Publishing, Cham, 2019. doi:https://doi.org/10.1007/978-3-319-91908-9_5.

agatz2018 [4] N. Agatz, P. Bouman, and M. Schmidt. Optimization Approaches for the Traveling Salesman Problem with Drone. *Transportation Science*, 52(4):965–981, 2018. doi:https://doi.org/10.1287/trsc.2017.0791.

akpakwu2018 [5] G. A. Akpakwu, G. P. Silva, B. J.and Hancke, and A. M. Abu-Mahfouz. A Survey on 5G Networks for the Internet of Things: Communication Technologies and Challenges. *IEEE Access*, 6:3619–3647, 2018. doi:https://doi.org/10.1109/ACCESS.2017.27798440.

hourani2014 [6] A. Al-Hourani, S. Kandeepan, and S. Lardner. Optimal LAP Altitude for Maximum Coverage. *IEEE Wireless Communications Letters*, 3(6):569–572, 2014. doi:http://dx.doi.org/10.1109/LWC.2014.2342736.

fadi2019 [7] F. Al-Turjman, J. P. Lemayian, S. Alturjman, and L. Mostarda. Enhanced Deployment Strategy for the 5G Drone-BS Using Artificial Intelligence. *IEEE Access*, 7:75999–76008, 2019. doi:http://dx.doi.org/10.1109/ACCESS.2019.2921729.

alzboon2017 [8] L. Alzboon, B. Khassawneh, and B. Nagy. On the number of weighted shortest paths in the square grid. In *2017 IEEE 21st International Conference on Intelligent Engineering Systems (INES)*, 2017. doi:https://doi.org/10.1109/INES.2017.8118533.

deoptim [9] D. Ardia, K. Mullen, B. Peterson, J. Ulrich, and K. Boudt. DEoptim: Global Optimization by Differential Evolution, 2016. https://cran.r-project.org/web/packages/DEoptim/index.html, online; accessed 12-September-2019.

arora1998 [10] S. Arora. Polynomial Time Approximation Schemes for Euclidean Traveling Salesman and Other Geometric Problems. *J. ACM*, 45(5):753–782, sep 1998. doi:https://doi.org/10.1145/290179.290180.

arshad2016 [11] R. Arshad, H. Elsawy, S. Sorour, T. Y. Al-Naffouri, and M. Alouini. Handover Management in 5G and Beyond: A Topology Aware Skipping Approach. *IEEE Access*, 4: 9073–9081, 2016. doi:https://doi.org/10.1109/ACCESS.2016.2642538.

saeed2017 [12] S. Asaeedi, F. Didehvar, and A. Mohades. α-Concave hull, a generalization of convex hull. *Theoretical Computer Science*, 702:48–59, 2017. doi:https://doi.org/10.1016/j.tcs.2017.08.014.

aurenhammer1991 [13] F. Aurenhammer. Voronoi Diagrams - a Survey of a Fundamental Geometric Data Structure. *ACM Comput. Surv.*, 23(3):345–405, 1991. doi:https://doi.org/10.1145/116873.116880.

convexhull2005 [14] A. Baddeley and R. Turner. spatstat: An R Package for Analyzing Spatial Point Patterns. *Journal of Statistical Software*, 12(6):1–42, 2005. doi:http://dx.doi.org/10.18637/jss.v012.i06.

bartholdi1988 [15] J. J. Bartholdi and L. K. Platzman. Heuristics Based on Spacefilling Curves for Combinatorial Problems in Euclidean Space. *Management Science*, 34(3):291–305, 1988. http://www.jstor.org/stable/2632046.

alphahull2019 [16] Beatriz Pateiro-Lopez and Alberto Rodriguez-Casal. *alphahull: Generalization of the Convex Hull of a Sample of Points in the Plane, R package version 2.2*, 2019. hhttps://CRAN.R-project.org/package=alphahull.

becker2017 [17] A. Becker, E. Fox-Epstein, P. N. Klein, and D. Meierfrankenfeld. Engineering an Approximation Scheme for Traveling Salesman in Planar Graphs. In Costas S. Iliopoulos and Solon P. Pissis and Simon J. Puglisi and Rajeev Raman, editor, *16th International Symposium on Experimental Algorithms (SEA 2017)*, volume 75, pages 8:1–8:17, 2017. doi:https://doi.org/10.4230/LIPIcs.SEA.2017.8.

binh2019 [18] H. T. T. Binh and N. H. Nam. Introduction to Coverage Optimization in WSNs. In H. T. T. Binh and N. Dey, editors, *Soft Computing in Wireless Sensor Networks*, chapter 6, pages 115–136. CRC Press, 2019.

borlaug2020 [19] B. Borlaug, S. Salisbury, M. Gerdes, and M. Muratori. Levelized Cost of Charging Electric Vehicles in the United States. *Joule*, 4(7):1470–1485, 2020. doi:https://doi.org/10.1016/j.joule.2020.05.013.

chazelle1985 [20] B. Chazelle. On the convex layers of a planar set. *IEEE Transactions on Information Theory*, 31(4):509–517, 1985. doi:https://doi.org/10.1109/TIT.1985.1057060.

ravi2016 [21] V. V. Chetlur Ravi and H. S. Dhillon. Downlink coverage probability in a finite network of UAV base stations. In *2016 IEEE 17th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, pages 1–5, July 2016. doi:http://dx.doi.org/10.1109/SPAWC.2016.7536859.

choi2016 [22] 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 *2016 International Conference on Control, Automation and Information Sciences (ICCAIS)*, pages 132–136, 2016. doi:https://doi.org/110.1109/ICCAIS.2016.7822448.

choi2021 [23] Y. Choi and P. M. Schonfeld. A comparison of optimized deliveries by drone and truck. *Transportation Planning and Technology*, 44(3):319–336, 2021. doi:https://doi.org/10.1080/03081060.2021.1883230.

christofides2021 [24] N. Christofides. Worst-Case Analysis of a New Heuristic for the Travelling Salesman Problem. *Operations Research Forum*, 3(2), 2021. doi:https://doi.org/10.1007/s43069-021-00101-z.

christofides1975 [25] N. Christofides and S. Korman. A Computational Survey of Methods for the Set Covering Problem. *Management Science*, 21(5):591–599, 1975. http://www.jstor.org/stable/2630042.

nestor2021 [26] N. M. Cid-Garcia and Y. A. Rios-Solis. Exact solutions for the 2d-strip packing problem using the positions-and-covering methodology. *PLOS ONE*, 16(1):1–20, 2021. doi:https://doi.org/10.1371/journal.pone.0245267.

claesson2020 [27] 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*, 156:196–201, 2020. doi:https://doi.org/10.1016/j.resuscitation.2020.09.022.

collins2003 [28] C. R. Collins and K. Stephenson. A circle packing algorithm. *Computational Geometry*, 25(3):233–256, 2003. doi:https://doi.org/10.1016/S0925-7721(02)00099-8.

croes1958 [29] G. A. Croes. A Method for Solving Traveling-Salesman Problems. *Operations Research*, 6(6):791–812, 1958. http://www.jstor.org/stable/167074.

cullenfrey1999 [30] A. C. Cullen and H. C. Frey. *Probabilistic Techniques in Exposure Assessment: A Handbook for Dealing with Variability and Uncertainty in Models and Inputs*. Plenum Press, USA, 1999.

daponte2019 [31] P. Daponte, L. D. Vito, L. Glielmo, L. Iannelli, D. Liuzza, F. Picariello, and G. Silano. A review on the use of drones for precision agriculture. *IOP Conference Series: Earth and Environmental Science*, 275(1):012022, 2019. doi:https://doi.org/10.1088/1755-1315/275/1/012022.

das2006 [32] G. K. Das, S. Das, S. C. Nandy, and B. P. Sinha. Efficient algorithm for placing a given number of base stations to cover a convex region. *Journal of Parallel and Distributed Computing*, 66(11):1353–1358, 2006. doi:https://doi.org/10.1016/j.jpdc.2006.05.004.

deBerg2008_1 [33] M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars. *Voronoi Diagrams*, pages 147–171. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008. doi:https://doi.org/10.1007/978-3-540-77974-2_7.

deBerg2008_2 [34] M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars. *Delaunay Triangulationss*, pages 191–218. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008. doi:https://doi.org/10.1007/978-3-540-77974-2_9.

deineko1994 [35] V. G. Deineko, R. van Dal, and G. Rote. The convex-hull-and-line traveling salesman problem: a solvable case. *Information Processing Letters*, 51(3):141–148, 1994. doi:https://doi.org/10.1016/0020-0190(94)00071-9.

fitdistrplus2015 [36] M. L. Delignette-Muller and C. Dutang. fitdistrplus: An R Package for Fitting Distributions. *Journal of Statistical Software*, 64(4):1–34, 2015. https://www.jstatsoft.org/v64/i04/.

desai2018 [37] R. R. Desai, R. B. Chen, and W. Armington. A Pattern Analysis of Daily Electric Vehicle Charging Profiles: Operational Efficiency and Environmental Impacts. *Journal of Advanced Transportation*, 2018, 2018. doi:https://doi.org/10.1155/2018/6930932.

amol2008 [38] A. Deshpande, S. Khuller, A. Malekian, and M. Toossi. Energy Efficient Monitoring in Sensor Networks. In E. S. Laber, C. Bornstein, L. T. Nogueira, and L. Faria, editors, *LATIN 2008: Theoretical Informatics*, pages 436–448, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg. doi:https://doi.org/10.1007/978-3-540-78773-0_38.

deineko1996 [39] V. G. Deĭneko and G. J. Woeginger. The Convex-hull-and-k-line Travelling Salesman Problem. *Information Processing Letters*, 59(6):295–301, 1996. doi:https://doi.org/10.1016/0020-0190(96)00125-1.

dillencourt1987 [40] M. B. Dillencourt. Traveling salesman cycles are not always subgraphs of Delaunay triangulations or of minimum weight triangulations. *Information Processing Letters*, 24 (5):339–342, 1987. doi:https://doi.org/10.1016/0020-0190(87)90160-8.

dillencourt1996 [41] M. B. Dillencourt. Finding Hamiltonian cycles in Delaunay triangulations is NP-complete. *Discrete Applied Mathematics*, 64(3):207–217, 1996. doi:https://doi.org/10.1016/0166-218X(94)00125-W.

dillencourt1993 [42] W. D. Dillencourt, Michael B.and Smith. A simple method for resolving degeneracies in Delaunay triangulations. In Lingas, Andrzej and Karlsson, Rolf and Carlsson, Svante, editor, *Automata, Languages and Programming, ICALP 1993. Lecture Notes in Computer Science* , volume 700, pages 177–188. Springer Berlin Heidelberg, 1993. doi:https://doi.org/10.1007/3-540-56939-1_71.

doddapaneni2018 [43] K. Doddapaneni, A. Tasiran, F. A. Omondi, E. Ever, P. Shah, L. Mostarda, and O. Gemikonakli. Does the assumption of exponential arrival distributions in wireless sensor networks hold? *International Journal of Sensor Networks (IJSNET)*, 26(2):81–100, 2018. doi:https://dx.doi.org/10.1504/IJSNET.2018.089258.

evmaintenance [44] DoE: Dept. of Energy (USA). Battery-Electric Vehicles Have Lower Scheduled Maintenance Costs than Other Light-Duty Vehicles. USA Dept. of Energy report, 2021. https://www.energy.gov/eere/vehicles/articles/fotw-1190-june-14-2021-battery-electric-vehicles-have-lower-scheduled, online; accessed: 18-March-2022.

domanski2005 [45] Z. Domański and J. Kęsy. Distribuition of Manhattan distance in square and triangular lattices. *Scientific Research of the Institute of Mathematics and Computer Science*, 4(1): 34–37, 2005. http://amcm.pcz.pl/get.php?article=2005_1/art_06.pdf.

dronelife [46] Dronelife. A Drone Battery That Charges in 5 Minutes. Dronlife news, 2021. https://dronelife.com/2020/08/01/a-drone-battery-that-charges-in-5-minutes/, online; accessed: 4-January-2021.

duckham2008 [47] 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 Recognition*, 41(10):3224–3236, 2008. doi:https://doi.org/10.1016/j.patcog.2008.03.023.

englert2014 [48] M. Englert, H. Röglin, and B. Vöcking. Worst Case and Probabilistic Analysis of the 2-Opt Algorithm for the TSP. *Algorithmica*, 68(1):190–264, 2014. doi:https://doi.org/10.1007/s00453-013-9801-4.

epaev [49] EPA: Environmental Protection Agency (USA). Electric Vehicle Myths. Green Vehicle Guide, 2022. https://www.epa.gov/greenvehicles/electric-vehicle-myths, online; accessed: 9-March-2022.

erdelj2017 [50] M. Erdelj, E. Natalizio, K. R. Chowdhury, and I. F. Akyildiz. Help from the Sky: Leveraging UAVs for Disaster Management. *IEEE Pervasive Computing*, 16(1):24–32, 2017. doi:https://doi.org/10.1109/MPRV.2017.11.

fenton16 [51] A. Fenton. The Bees Algorithm for the Vehicle Routing Problem. Master's thesis, Department of Computer Science, University of Auckland, 2016. http://arxiv.org/abs/1605.05448.

flood1956 [52] M. M. Flood. The Traveling-Salesman Problem. *Operations Research*, 4(1):61–75, 1956. http://www.jstor.org/stable/167517.

hazmat [53] flytnow. Role of Drones and FlytNow in HAZMAT Response. flytnow news, 2022. https://flytnow.com/drone-for-hazmat-response/, online; accessed: 13-April-2022.

formica2021 [54] N. Formica, L. Mostarda, and A. Navarra. UAVs Route Planning in Sea Emergencies. In L. Barolli, I. Woungang, and T. Enokido, editors, *Advanced Information Networking and Applications*, pages 588–599, Cham, 2021. Springer International Publishing. doi:https://doi.org/10.1007/978-3-030-75100-5_51.

fotouhi2019 [55] A. Fotouhi, H. Qiang, M. Ding, M. Hassan, L. G. Giordano, A. García-Rodríguez, and J. Yuan. Survey on UAV Cellular Communications: Practical Aspects, Standardization Advancements, Regulation, and Security Challenges. *IEEE Communications Surveys Tutorials*, pages 1–1, 2019. doi:http://dx.doi.org/10.1109/COMST.2019.2906228.

galkin2019 [56] B. Galkin, J. Kibilda, and L. A. DaSilva. UAVs as Mobile Infrastructure: Addressing Battery Lifetime. *IEEE Communications Magazine*, 57(6):132–137, 2019. doi:https://doi.org/10.1109/MCOM.2019.1800545.

galton2006 [57] A. Galton and M. Duckham. What Is the Region Occupied by a Set of Points? In Raubal, Martin and Miller, Harvey J. and Frank, Andrew U. and Goodchild, Michael F., editor, *Geographic Information Science*, pages 81–98. Springer Berlin Heidelberg, 2006. doi:https://doi.org/10.1007/11863939_6.

gilli2019 [58] M. Gilli, D. Maringer, and E. Schumann. *Numerical Methods and Optimization in Finance*. Academic Press, 2nd edition, 2019. doi:https://doi.org/10.1016/B978-0-12-815065-8.00024-8.

glock2020 [59] K. D. Glock. *Emergency rapid mapping with drones Models and solution approaches for offline and online mission planning*. PhD thesis, Faculty of Economics at the Karlsruhe Institute of Technology (KIT), Germany, 2020. https://publikationen.bibliothek.kit.edu/1000124518/90282175.

concaveman2020 [60] J. Gombin, R. Vaidyanathan, and V. Agafonkin. *concaveman: A Very Fast 2D Concave Hull Algorithm, R package version 1.1.0*, 2020. https://CRAN.R-project.org/package=concaveman.

guo2020 [61] 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 Engineering*, 216:107693, 2020. doi:https://doi.org/10.1016/j.oceaneng.2020.107693.

punnen2007 [62] G. Gutin and A.-P. Punnen. *The Traveling Salesman Problem and Its Variations*. Springer-Verlag US, Boston, MA, 2007. doi:https://doi.org/10.1007/b101971.

hafez2022 [63] A. Hafeez, M. A. Husain, S. Singh, A. Chauhan, M. T. Khan, N. Kumar, A. Chauhan, and S. Soni. Implementation of drone technology for farm monitoring & pesticide spraying: A review. *Information Processing in Agriculture*, 2022. doi:https://doi.org/10.1016/j.inpa.2022.02.002.

tsp2007 [64] M. Hahsler and K. Hornik. TSP – Infrastructure for the traveling salesperson problem. *Journal of Statistical Software*, 23(2):1–21, 2007. doi:https://doi.org/10.18637/jss.v023.i02.

harpeled2011 [65] S. Har-Peled. *Geometric Approximation Algorithms*. American Mathematical Society, 2011.

hassanalian2017 [66] M. Hassanalian and A. Abdelkefi. Classifications, applications, and design challenges of drones: A review. *Progress in Aerospace Sciences*, 91:99–131, 2017. doi:https://doi.org/10.1016/j.paerosci.2017.04.003.

hernandez2013 [67] S. A. Hernández, G. Leguizamón, and E. Mezura-Montes. Hybridization of Differential Evolution using Hill Climbing to solve Constrained Optimization Problems. *Inteligencia Artif.*, 16(52):3–15, 2013. http://journal.iberamia.org/index.php/ia/article/view/1032/article%20%281%29.pdf.

hougardy2021 [68] S. Hougardy and X. Zhong. Hard to solve instances of the Euclidean Traveling Salesman Problem. *Mathematical Programming Computation*, 13:51—74, 2021. doi:https://doi.org/10.1007/s12532-020-00184-5.

huang2020 [69] H. Huang and A. V. Savkin. A Method of Optimized Deployment of Charging Stations for Drone Delivery. *IEEE Transactions on Transportation Electrification*, 6(2):510–518, 2020. doi:https://doi.org/10.1109/TTE.2020.2988149.

huang2021 [70] H. Huang and A. V. Savkin. Optimal Deployment of Charging Stations for Aerial Surveillance by UAVs with the Assistance of Public Transportation Vehicles. *Sensors*, 21(16):5320, 2021. doi:https://doi.org/10.3390/s21165320.

ibrahim2006 [71] W. Ibrahim, A. El-Chouemi, and H. El-Sayed. Novel Heuristic and Genetic Algorithms for the VLSI Test Coverage Problem. In *IEEE International Conference on Computer Systems and Applications, 2006.*, pages 402–408, 2006. doi:https://doi.org/10.1109/AICCSA.2006.205122.

tictoc2021 [72] S. Izrailev. *tictoc: Functions for Timing R Scripts, as Well as Implementations of Stack and List Structures, R package version 1.0.1*, 2021. https://CRAN.R-project.org/package=tictoc.

jones2014 [73] J. Jones and A. Adamatzky. Computation of the travelling salesman problem by a shrinking blob. *Natural Computing*, 13(1):1–16, 2014. doi:https://doi.org/10.1007/s11047-013-9401-x.

ju2011 [74] L. Ju, T. Ringler, and M. Gunzburger. Voronoi Tessellations and Their Application to Climate and Global Modeling. In P. Lauritzen, C. Jablonowski, M. Taylor, and R. Nair, editors, *Numerical Techniques for Global Atmospheric Models*, pages 313–342. Springer Berlin Heidelberg, 2011. doi:https://doi.org/10.1007/978-3-642-11640-7_10.

junger1995 [75] M. Jünger, G. Reinelt, and G. Rinaldi. Chapter 4: The traveling salesman problem. In M.O. Ball, T.L. Magnanti, C.L. Monma and G.L. Nemhauser, editor, *Network Models*, volume 7 of *Handbooks in Operations Research and Management Science*, pages 225–330. Elsevier, 1995. doi:https://doi.org/10.1016/S0927-0507(05)80121-5.

kabbabe2021 [76] K. Kabbabe Poleo, W. J. Crowther, and M. Barnes. Estimating the impact of drone-based inspection on the Levelised Cost of electricity for offshore wind farms. *Results in Engineering*, 9:100201, 2021. doi:https://doi.org/10.1016/j.rineng.2021.100201.

kantabutra1983 [77] V. Kantabutra. Traveling salesman cycles are not always subgraphs of Voronoi duals. *Information Processing Letters*, 16(1):11–12, 1983. doi:https://doi.org/10.1016/0020-0190(83)90004-2.

karaca2018 [78] 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. *The American Journal of Emergency Medicine*, 36(4):583–588, 2018. doi:https://doi.org/10.1016/j.ajem.2017.09.025.

karp1972 [79] R. M. Karp. Reducibility among Combinatorial Problems. In R. E. Miller, J. W. Thatcher, and J. D. Bohlinger, editors, *Complexity of Computer Computations: Proceedings of a symposium on the Complexity of Computer Computations, held March 20–22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, and sponsored by the Office of Naval Research, Mathematics Program, IBM World Trade Corporation, and the IBM Research Mathematical Sciences Department*, pages 85–103. Springer US, Boston, MA, 1972. doi:https://doi.org/10.1007/978-1-4684-2001-2_9.

katoch2021 [80] S. Katoch, S. S. Chauhan, and V. Kumar. A review on genetic algorithm: past, present, and future. *Multimedia Tools and Applications*, 80(5):8091–8126, 2021. doi:https://doi.org/10.1007/s11042-020-10139-6.

khanimov2015 [81] M. Khanimov and M. Sharir. Delaunay Triangulations of Degenerate Point Sets. *CoRR*, abs/1510.04608, 2015. http://arxiv.org/abs/1510.04608.

khoufi2019 [82] 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*, 3(3):66, 2019. doi:https://doi.org/10.3390/drones3030066.

samir-vrp [83] S. Khuller, A. Malekian, and J. Mestre. To Fill or Not to Fill: The Gas Station Problem. *ACM Trans. Algorithms*, 7(3):36:1–36:16, 2011. doi:https://doi.org/10.1145/1978782.1978791.

kilic2021_2 [84] K. I. Kilic and L. Mostarda. Optimum Path Finding Framework for Drone Assisted Boat Rescue Missions. In Barolli, Leonard and Woungang, Isaac and Enokido, Tomoya, editor, *Advanced Information Networking and Applications. AINA 2021. Lecture Notes in Networks and Systems*, volume 227, pages 219–231. Springer International Publishing, 2021. doi:https://doi.org/10.1007/978-3-030-75078-7_23.

kilic2021_3 [85] K. I. Kilic and L. Mostarda. Heuristic Drone Pathfinding Over Optimized Charging Station Grid. *IEEE Access*, 9:164070–164089, 2021. doi:https://doi.org/10.1109/ACCESS.2021.3134459.

kilic2022 [86] K. I. Kilic and L. Mostarda. Novel Concave Hull-Based Heuristic Algorithm For TSP. *Operations Research Forum*, 3(2):25, 2022. doi:https://doi.org/10.1007/s43069-022-00137-9.

kilic2020 [87] K. I. Kilic, O. Gemikonakli, and L. Mostarda. Multi-objective Priority Based Heuristic Optimization for Region Coverage with UAVs. In L. Barolli, F. Amato, F. Moscato,

T. Enokido, and M. Takizawa, editors, *Advanced Information Networking and Applications*, pages 768–779. Springer International Publishing, 2020. ISBN 978-3-030-44041-1. doi:https://doi.org/10.1007/978-3-030-44041-1_68.

kilic2021_1 [88] 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. *International Journal of Web and Grid Services*, 17(2):152–178, 2021. doi:https://doi.org/10.1504/IJWGS.2021.114574.

kirckpatrick1983 [89] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by Simulated Annealing. *Science*, 220(4598):671–680, 1983. doi:https://doi.org/10.1126/science.220.4598.671.

kuru2021 [90] K. Kuru. Planning the Future of Smart Cities With Swarms of Fully Autonomous Unmanned Aerial Vehicles Using a Novel Framework. *IEEE Access*, 9:6571–6595, 2021. doi:https://doi.org/10.1109/ACCESS.2020.3049094.

letchford2008 [91] A. N. Letchford and N. A. Pearson. Good triangulations yield good tours. *Computers & Operations Research*, 35(2):638–647, 2008. Part Special Issue: Location Modeling Dedicated to the memory of Charles S. ReVelle.doi:https://doi.org/10.1016/j.cor.2006.03.025.

leung2004 [92] J. Leung. *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*. Chapman & Hall/CRC Computer and Information Science Series. CRC Press, 2004.

li2019 [93] B. Li, Z. Fei, and Y. Zhang. UAV Communications for 5G and Beyond: Recent Advances and Future Trends. *IEEE Internet of Things Journal*, 6(2):2241–2263, 2019. doi:http://dx.doi.org/10.1109/JIOT.2018.2887086.

li2015 [94] X. Li, D. Guo, H. Yin, and G. Wei. Drone-assisted public safety wireless broadband network. In *2015 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, pages 323–328, March 2015. doi:http://dx.doi.org/10.1109/WCNCW.2015.7122575.

lian2009 [95] S. Lianshuan and L. Zengyan. An Improved Pareto Genetic Algorithm for Multi-objective TSP. In *2009 Fifth International Conference on Natural Computation*, volume 4, pages 585–588, 2009. doi:https://doi.org/10.1109/ICNC.2009.510.

lin2016 [96] Y. Lin, Z. Bian, and X. Liu. Developing a dynamic neighborhood structure for an adaptive hybrid simulated annealing – tabu search algorithm to solve the symmetrical traveling salesman problem. *Applied Soft Computing*, 49:937–952, 2016. doi:https://doi.org/10.1016/j.asoc.2016.08.036.

lust2010 [97] T. Lust and J. Teghem. The Multiobjective Traveling Salesman Problem: A Survey and a New Approach. In C. A. Coello Coello, C. Dhaenens, and L. Jourdan, editors, *Advances in Multi-Objective Nature Inspired Computing*, pages 119–141. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010. doi:https://doi.org/10.1007/978-3-642-11218-8_6.

lyu2018 [98] J. Lyu, Y. Zeng, and R. Zhang. UAV-Aided Offloading for Cellular Hotspot. *IEEE Transactions on Wireless Communications*, 17(6):3988–4001, 2018. doi:http://dx.doi.org/110.1109/TWC.2018.2818734.

maini2015 [99] P. Maini and P. B. Sujit. On cooperation between a fuel constrained UAV and a refueling UGV for large scale mapping applications. In *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1370–1377, 2015. doi:https://doi.org/10.1109/ICUAS.2015.7152432.

manthey2009 [100] B. Manthey and L. Shankar Ram. Approximation Algorithms for Multi-Criteria Traveling Salesman Problems. *Algorithmica*, 53(1):69–88, 2009. doi:https://doi.org/10.1007/s00453-007-9011-z.

marinei [101] Marinei. The growing role for aerial drones in the maritime industry. Marinei news, 2020. https://www.marine-i.co.uk/news/article/80/the-growing-role-for-aerial-drones-in-the-maritime-industry, online; accessed: 29-December-2020.

matai2010 [102] R. Matai, S. P. Singh, and M. L. Mittal. Chapter 1: Traveling salesman problem: An Overview Of Applications, Formulations, And Solution Approaches. In Donald Davendra, editor, *Traveling Salesman Problem: Theory and Applications*. IntechOpen, 2010. doi:https://doi.org/10.5772/12909.

mbiadou2018 [103] 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*, 72(4):459–474, 2018. doi:https://doi.org/10.1002/net.21846.

raissa2021 [104] R. G. Mbiadou Saleu, L. Deroussi, D. Feillet, N. Grangeon, and A. Quilliot. The parallel drone scheduling problem with multiple drones and vehicles. *European Journal of Operational Research*, 2021. doi:https://doi.org/10.1016/j.ejor.2021.08.014.

mcrea2019 [105] J. N. McRae, J. C. 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 & Environmental Medicine*, 30(3):287–290, 2019. doi:https://doi.org/10.1016/j.wem.2019.03.004.

merwaday2016 [106] A. Merwaday, A. Tuncer, A. Kumbhar, and I. Guvenc. Improved Throughput Coverage in Natural Disasters: Unmanned Aerial Base Stations for Public-Safety Communications. *IEEE Vehicular Technology Magazine*, 11(4):53–60, 2016. doi:https://doi.org/10.1109/MVT.2016.2589970.

michalewicz1996 [107] Z. Michalewicz. *Genetic algorithms + data structures = evolution programs (3rd, extended ed.)*. Springer, Berlin, Heidelberg, 1996. doi:https://doi.org/10.1007/978-3-662-03315-9.

michini2011 [108] 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 *Infotech@Aerospace 2011*, 2011. doi:http://dx.doi.org/10.2514/6.2011-1405.

mishra2020 [109] B. Mishra, D. Garg, P. Narang, and V. Mishra. Drone-surveillance for search and rescue in natural disaster. *Computer Communications*, 156:1–10, 2020. doi:https://doi.org/10.1016/j.comcom.2020.03.012.

moreira2007 [110] 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 *GRAPP 2007 - International Conference on Computer Graphics Theory and Applications*, pages 61–68, 2007.

mozaffari2019 [111] M. Mozaffari, W. Saad, M. Bennis, Y.-H. Nam, and M. Debbah. A Tutorial on UAVs for Wireless Networks: Applications, Challenges, and Open Problems. *IEEE Communications Surveys Tutorials*, pages 1–1, 2019. doi:http://dx.doi.org/10.1109/COMST.2019.2902862.

mullen2014 [112] K. Mullen. Continuous Global Optimization in R. *Journal of Statistical Software, Articles*, 60(6):1–45, 2014. doi:http://dx.doi.org/10.18637/jss.v060.i06.

murray2015 [113] C. C. Murray and A. G. Chu. The flying sidekick traveling salesman problem: Optimization of drone-assisted parcel delivery. *Transportation Research Part C: Emerging Technologies*, 54:86–109, 2015. doi:https://doi.org/10.1016/j.trc.2015.03.005.

narasimhan2007 [114] G. Narasimhan and M. Smid. *Geometric Spanner Networks*. Cambridge University Press, 2007. doi:https://doi.org/10.1017/CBO9780511546884.

parham2021 [115] P. Nooralishahi, C. Ibarra-Castanedo, S. Deane, F. López, S. Pant, M. Genest, N. P. Avdelidis, and X. P. V. Maldague. Drone-Based Non-Destructive Inspection of Industrial Sites: A Review and Case Studies. *Drones*, 5(4), 2021. doi:https://doi.org/10.3390/drones5040106.

okabe2000 [116] A. Okabe, B. Boots, K. Sugihara, S. N. Chiu, and D. G. Kendall. *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. Wiley, 2^{nd} edition, 2000. doi:https://doi.org/10.1002/9780470317013.

ouyang2018 [117] J. Ouyang, Y. Che, J. Xu, and K. Wu. Throughput Maximization for Laser-Powered UAV Wireless Communication Systems. In *2018 IEEE International Conference on Communications Workshops (ICC Workshops)*, pages 1–6, 2018. doi:https://doi.org/10.1109/ICCW.2018.8403572.

palatinus2010 [118] E. Palatinus and B. Bánhelyi. Circle Covering and its Applications for Telecommunication Networks. In *Proceedings of the 8th International Conference on Applied Informatics*, pages 255—262, January 2010. http://icai.ektf.hu/pdf/ICAI2010-vol2-pp255-262.pdf.

park2013 [119] J.-S. Park and S.-J. Oh. A new concave hull algorithm and concaveness measure for n-dimensional datasets. *Journal of Information Science and Engineering*, 29(2):379–392, 2013.

bartholdi1989 [120] L. K. Platzman and J. J. Bartholdi. Spacefilling Curves and the Planar Travelling Salesman Problem. *Journal of the ACM*, 36(4):719–737, 1989. doi:https://doi.org/10.1145/76359.76361.

price2005 [121] K. Price, R. M. Storn, and J. A. Lampinen. *Differential Evolution: A Practical Approach to Global Optimization*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005. doi:https://doi.org/10.1007/3-540-31306-0.

punnen2003 [122] Punnen, Margot, and Kabadi. TSP Heuristics: Domination Analysis and Complexity. *Algorithmica*, 35(2):111–127, 2003. doi:https://doi.org/10.1007/s00453-002-0986-1.

ramaswami1993 [123] S. Ramaswami. Convex Hulls: Complexity and Applications (a Survey). Technical report, Department of Computer & Information Science, University of Pennsylvania, 1993. https://repository.upenn.edu/cgi/viewcontent.cgi?article=1272&context=cis_reports, Online; accessed 31-March-2022.

rivera2017 [124] N. Rivera, C. Hernández, and J. Baier. Grid Pathfinding on the 2k Neighborhoods. In *AAAI Conference on Artificial Intelligence*, 2017. https://www.aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/15014.

roberti2019 [125] R. Roberti and M. Ruthmair. Exact Methods for the Traveling Salesman Problem with Drone. *Optimization Online*, 2019. http://www.optimization-online.org/DB_FILE/2019/11/7457.pdf.

roberti2021 [126] R. Roberti and M. Ruthmair. Exact Methods for the Traveling Salesman Problem with Drone. *Transportation Science*, 55(2):315–335, 2021. doi:https://doi.org/10.1287/trsc.2020.1017.

rosenkrantz2009 [127] D. J. Rosenkrantz, R. E. Stearns, and P. M. Lewis. An analysis of several heuristics for the traveling salesman problem. In Ravi, S. S. and Shukla, Sandeep K., editor, *Fundamental Problems in Computing: Essays in Honor of Professor Daniel J. Rosenkrantz*, pages 45–69. Springer Netherlands, Dordrecht, 2009. doi:https://doi.org/10.1007/978-1-4020-9688-4_3.

safar2009 [128] M. Safar, D. Ebrahimi, and D. Taniar. Voronoi-based reverse nearest neighbor query processing on spatial networks. *Multim. Syst.*, 15(5):295–308, 2009. doi:https://doi.org/10.1007/s00530-009-0167-z.

ga [129] L. Scrucca. GA: Genetic Algorithms, 2019. https://cran.r-project.org/web/packages/GA/index.html, online; accessed 12-September-2019.

sekander2018 [130] S. Sekander, H. Tabassum, and E. Hossain. Multi-Tier Drone Architecture for 5G/B5G Cellular Networks: Challenges, Trends, and Prospects. *IEEE Communications Magazine*, 56(3):96–103, 2018. doi:https://doi.org/10.1109/MCOM.2018.1700666.

shak2019 [131] 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*, 7:48572–48634, 2019. doi:https://doi.org/10.1109/ACCESS.2019.2909530.

sharaf2019 [132] S. Sharafeddine and R. Islambouli. On-demand deployment of multiple aerial base stations for traffic offloading and network recovery. *Computer Networks*, 156:52–61, 2019. doi:https://doi.org/10.1016/j.comnet.2019.03.016.

silvagni2017 [133] M. Silvagni, A. Tonoli, E. Zenerino, and M. Chiaberge. Multipurpose UAV for search and rescue operations in mountain avalanche events. *Geomatics, Natural Hazards and Risk*, 8(1):18–33, 2017. doi:https://doi.org/10.1080/19475705.2016.1238852.

diffevo [134] R. M. Storn. Differential Evolution (DE) for Continuous Function Optimization (an algorithm by Kenneth Price and Rainer Storn), 2019. http://www1.icsi.berkeley.edu/~storn/code.html, online; accessed 31-October-2019.

sundar2014 [135] K. Sundar and S. Rathinam. Algorithms for Routing an Unmanned Aerial Vehicle in the Presence of Refueling Depots. *IEEE Transactions on Automation Science and Engineering*, 11(1):287–294, 2014. doi:https://doi.org/10.1109/TASE.2013.2279544.

tang2019 [136] Z. Tang, W.-J. v. Hoeve, and P. Shaw. A Study on the Traveling Salesman Problem with a Drone. In L.-M. Rousseau and K. Stergiou, editors, *Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, pages 557–564, Cham, 2019. Springer International Publishing. doi:https://doi.org/10.1007/978-3-030-19212-9_37.

dot2013 [137] US Department of Transportation. Unmanned Aircraft System (UAS) Service Demand 2015–2035: Literature Review & Projections of Future Usage, Version 0.1. Technical report, US Department of Transportation, 2013. https://www.hsdl.org/?view&did=749449, Online; accessed 16-October-2019.

vanleeuwen2009 [138] E. J. van Leeuwen. *Optimization and approximation on systems of geometric objects*. PhD thesis, Faculty of Science (FNWI), Korteweg-de Vries Institute for Mathematics (KdVI), Netherlands, 2009. https://hdl.handle.net/11245/1.307107.

vanderbei2014 [139] R. J. Vanderbei. *Integer Programming*, pages 345–362. Springer US, Boston, MA, 2014. doi:https://doi.org/10.1007/978-1-4614-7630-6_23.

vazirani2003 [140] V. V. Vazirani. *Approximation Algorithms*. Springer Berlin Heidelberg, 2003. doi:https://doi.org/10.1007/978-3-662-04565-7.

vasq2021 [141] S. A. Vásquez, G. Angulo, and M. A. Klapp. An exact solution method for the TSP with Drone based on decomposition. *Computers & Operations Research*, 127:105127, 2021. doi:https://doi.org/10.1016/j.cor.2020.105127.

wang2016 [142] J. Wang, O. K. Ersoy, M. He, and F. Wang. Multi-offspring genetic algorithm and its application to the traveling salesman problem. *Applied Soft Computing*, 43:415–423, 2016. doi:https://doi.org/10.1016/j.asoc.2016.02.021.

gensa [143] Y. Xiang, S. Gubian, B. Suomela, and J. Hoeng. GenSA: Generalized Simulated Annealing, 2019. https://cran.r-project.org/web/packages/GenSA/index.html, online; accessed 12-September-2019.

xuan20111 [144] K. Xuan, G. Zhao, D. Taniar, J. W. Rahayu, M. Safar, and B. Srinivasan. Voronoi-based range and continuous range query processing in mobile databases. *J. Comput. Syst. Sci.*, 77(4):637–651, 2011. doi:https://doi.org/10.1016/j.jcss.2010.02.005.

xuan20112 [145] K. Xuan, G. Zhao, D. Taniar, M. Safar, and B. Srinivasan. Voronoi-based multi-level range search in mobile navigation. *Multim. Tools Appl.*, 53(2):459–479, 2011. doi:https://doi.org/10.1007/s11042-010-0498-y.

yin2019 [146] S. Yin, Y. Zhao, and L. Li. Resource Allocation and Basestation Placement in Cellular Networks With Wireless Powered UAVs. *IEEE Transactions on Vehicular Technology*, 68(1):1050–1055, 2019. doi:https://doi.org/10.1109/TVT.2018.2883093.

younis2008 [147] M. Younis and K. Akkaya. Strategies and techniques for node placement in WSNs: A survey. *Ad Hoc Networks*, 6(4):621–655, 2008. doi:https://doi.org/10.1016/j.adhoc.2007.05.003.

yun2010 [148] Z. Yun, X. Bai, D. Xuan, T. H. Lai, and W. Jia. Optimal Deployment Patterns for Full Coverage and k-Connectivity ($k \leq 6$) WSNs. *IEEE/ACM Transactions on Networking*, 18(3):934–947, 2010. doi:http://dx.doi.org/10.1109/TNET.2010.2040191.

zeng2016 [149] Y. Zeng, R. Zhang, and T. J. Lim. Wireless communications with unmanned aerial vehicles: opportunities and challenges. *IEEE Communications Magazine*, 54(5):36–42, 2016. doi:http://dx.doi.org/10.1109/MCOM.2016.7470933.

zhang2019 [150] H. Zhang, L. Song, and Z. Han. *Unmanned Aerial Vehicle Applications over Cellular Networks for 5G and Beyond*. Springer International Publishing, 2019. doi:https://doi.org/10.1007/978-3-030-33039-2.

zhang2012 [151] L. Zhang, D. Li, H. Zhu, and L. Cui. OPEN: An optimisation scheme of N-node coverage in WSNs. *IET Wireless Sensor Systems*, 2(1):40–51, 2012. doi:http://dx.doi.org/10.1049/iet-wss.2011.0012.

zhao2011 [152] G. Zhao, K. Xuan, J. W. Rahayu, D. Taniar, M. Safar, M. L. Gavrilova, and B. Srinivasan. Voronoi-Based Continuous k Nearest Neighbor Search in Mobile Navigation. *IEEE Trans. Ind. Electron.*, 58(6):2247–2257, 2011. doi:https://doi.org/10.1109/TIE.2009.2026372.

zhou2019 [153] A.-H. Zhou, L.-P. Zhu, B. Hu, S. Deng, Y. Song, H. Qiu, and S. Pan. Traveling-Salesman-Problem Algorithm Based on Simulated Annealing and Gene-Expression Programming. *Information*, 10(1), 2019. doi:https://doi.org/10.3390/info10010007.

zorbas2016 [154] D. Zorbas, L. D. P. Pugliese, T. Razafindralambo, and F. Guerriero. Optimal drone placement and cost-efficient target coverage. *Journal of Network and Computer Applications*, 75:16–31, 2016. doi:https://doi.org/10.1016/j.jnca.2016.08.009.