Zoning under Environmental Uncertainty

With Applications for Autonomous Vehicle Routing

Casper Bak Pedersen & Kasper Rosenkrands
Department of Mathematical Sciences

Master's Thesis

Mathematical Sciences
Aalborg University
http://www.aau.dk

AALBORG UNIVERSITY
STUDENT REPORT

Title:
Zoning under Environmental Uncertainty

Theme:
Autonomous Vehicle Routing

Project Period:
Spring Semester 2022

Project Group:
math-22-matoek-10-kaspercasper

Participant(s):
Casper Bak Pedersen
Kasper Rosenkrands

Supervisor(s):
Inkyung Sung
Peter Nielsen

Copies: 1

Page Numbers: 45

Date of Completion:
June 2, 2022

Abstract:

With the increasing size of wildfires (National Interagency Fire Center [21]) a need for effective ways of monitoring and containing the fires spread has emerged. The use of unmanned aerial vehicles (UAVs) for optical monitoring of a spreading wildfire is appropriate due to e.g., the longevity of missions and the hazardous environment.
With the increasing size of fires more UAVs are needed to effectively monitor an area, leading to more potential conflicts between UAVs, thereby requiring the need for communication if a UAV were to deviate from its planned route. However there are many situations where communication between UAVs is not practical or even impossible. This project therefore explores zoning methods that will allow each agent to have navigational freedom while not needing to communicate in order to avoid conflicts.
This problem is formulated as a 2-stage stochastic linear programming problem, that reflects the zoning and flight stage, where the environmental uncertainty during flight should be considered when constructing zones. This is solved using two zoning methods, inspired by the literature (Khemakhem et al. [18]) as well as a novel routing-based clustering method proposed by the authors. These are then compared using cumulative route scores for routes generated in the resulting zones. Their applicability for dynamic environments is further tested by considering how often each of the zoning solutions routes can be optimally updated while respecting zone boundaries.
Lastly, the aforementioned zoning methods are compared with a traditional routing approach, in order to gauge performance degradation due to zone restrictions.

Reading Guide

This chapter will give an overview of the reports structure, in order to show our progression through the process and introduce concepts in an intuitive order.

- Chapter 1 *Introduction*:

  This chapter gives context to the project, both in terms of practical application and related literature.

- Chapter 2 *Problem Formulation*:

  In this chapter a description of the problem considered and how to represent it is given, first conceptually and then mathematically.

- Chapter 3 *Solution Methods*:

  This chapter introduces the methods used to generate solutions and the considerations regarding each of them.

- Chapter 4 *Experiments*:

  In this chapter the solution methods presented in Chapter 3 will be tested on a modified set of test instances commonly seen in related literature.

- Chapter 5 *Discussion*:

  This chapter discusses the central choices made in the project and provides alternative solution methods and assumptions along with their predicted relevance and influence on the results obtained in this project.

- Chapter 6 *Conclusion*:

  Answers to the questions raised in the statement of intent and gives a summary of the central results from each chapter.

Contents

Preface **ix**

1 Introduction **1**
1.1 Literature Review . . . 1
1.2 Statement of Intent . . . 3

2 Problem Formulation **5**
2.1 Mathematical Formulation . . . 6
2.1.1 Dynamic Aspects . . . 7
2.1.2 The Zoned Team Orienteering Problem . . . 8
2.1.3 Problem Instances . . . 10

3 Solution Methods **11**
3.1 Zoning Approaches . . . 12
3.1.1 Heuristic-based Approaches . . . 12
3.1.2 Routing-based Approach . . . 13
3.2 Routing . . . 18
3.2.1 Pre-mission . . . 20
3.2.2 During-mission . . . 23

4 Experiments **27**
4.1 Experiment Design . . . 27
4.2 Comparison of Zoning Solution Methods . . . 27
4.2.1 Cumulative route score . . . 28
4.2.2 Routing flexibility within the cluster . . . 31
4.3 Comparison with the Team Orienteering Problem . . . 32
4.4 Reserving Range for Updating Routes . . . 34
4.5 Conclusion . . . 36

5 Discussion **37**

6 Conclusion **41**

Bibliography **43**

Preface

This Master's thesis is written by Casper Bak Pedersen and Kasper Rosenkrands, at The Faculty of Engineering and Science at Aalborg University.

The authors have used the following tools in the writing of this report

- **Overleaf** - collaborative writing with LaTeX,
- **Microsoft Teams** - virtual meetings and file sharing.

For implementation the following tools have been used

- **R** (4.1.1) - implementation of algorithms, simulation and data visualization.
  In regards to the implementation the main packages used are igraph (Csardi and Nepusz [11]) and the Tidyverse (Wickham et al. [26]).

Aalborg University, June 2, 2022

Casper Bak Pedersen
<cbpe17@student.aau.dk>

Kasper Rosenkrands
<krosen17@student.aau.dk>

1 Introduction

Forest fires have seen an increase in both scale and severity over the last decades. In Figure 1.1 data from National Interagency Fire Center [21] shows the change in number of wildfires and annual acres burned in the U.S., in the period from 1983 until 2021. The data shows a slight negative trend in the number of fires in the last two decades. However when looking at the area burned a sharp increase is evident in the period 2000-2010 and the following decade have stayed at this level. This suggests that fires on average have doubled in size over the course of a decade. Incidents with many forest fires in close proximity covering a million acres have been dubbed mega fires and have received particular attention given their destructive nature and related consequences such a tornadoes and thunderstorms [5]. Their scale makes them hard to monitor and therefore to effectively tame or extinguish and the data suggests that they happen more frequently than ever.

The goal of this project will therefore be to find methods for efficiently monitoring forest fires. Important factors to consider for this case are that the monitoring devices can come in harms way and the origin of the fire may only be determined as it becomes more severe. We therefore wish to use vehicles that can move fast and without risking direct contact with the fire. For optical monitoring the only real option is therefore aerial vehicles. Given that these mission can be very long and have a high likelihood of extreme heat, we propose the use of unmanned aerial vehicles (UAVs), as also seen in e.g. Casbeer et al. [7].

The problem then becomes designing routes for these UAVs to follow, such that as much useful information is gathered, while keeping the UAVs themselves relatively safe. Using graph theory this can be formulated as a generalization of the Travelling Salesman Problem (TSP) and different formulations from the literature will therefore be explored, in order to find generalizations that fit with the use case described above.

1.1 Literature Review

One of the most well known combinatorial optimization problems in graph theory and Operations Research in general is the TSP. The graph theoretical formulation of the problem is to find a Hamiltonian cycle with least weight in a complete weighted graph. That is, to find the shortest path with the same source and sink, that visits all nodes in the graph. Many generalizations of the TSP have evolved to suit a multitude of different applications. A subset of these generalizations from the literature on the subject will therefore be reviewed.

Firstly a way to evaluate how important or dangerous a given area is to visit, should be decided to give a basis for determining optimal routing. Laporte and Martello [19] consider the Selective TSP (STSP) where vertices of the graph have an associated profit. The objective is to find a circuit that maximises total profit while having a length not exceeding a pre-specified bound. This variant is also refered to as the Orienteering Problem (OP) by Tsiligirides [24].

The Vehicle Routing Problem (VRP) is the generalization of the TSP to multiple vehicles. Generalizing the VRP to include vertices with profits, and the objective of maximizing total profits within the

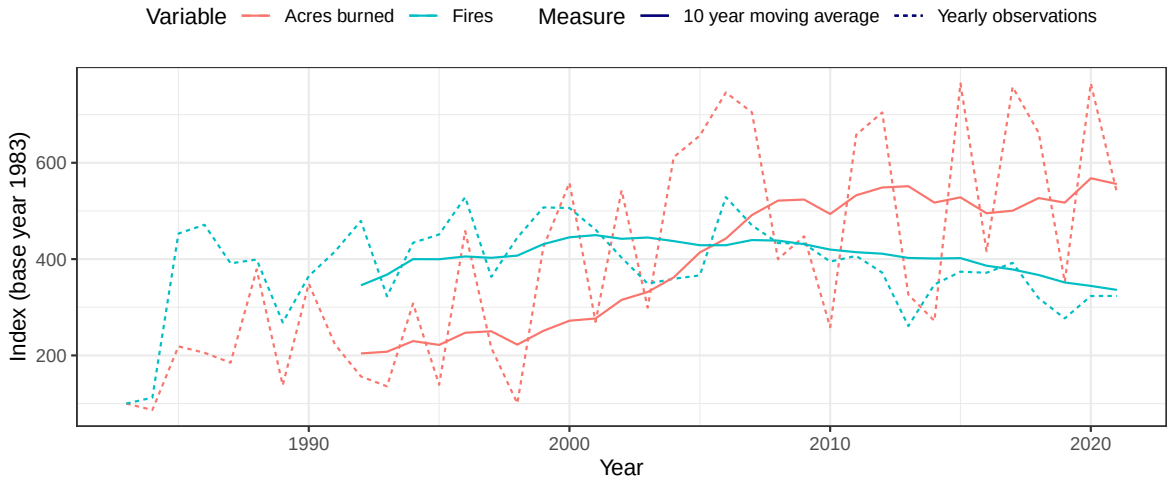


Figure 1.1: Development in number of wildfires and acres burned in the U.S., in the time period from 1983 until 2021, according to National Interagency Fire Center [21].

distance constraint, is introduced by Chao et al. [8] as the Team Orienteering problem (TOP). The same problem is also refered to as the VRP with Profits (VRPP) and the Selective VRP (SVRP) [18]. These generalizations to multiple agents are essential for the size of forest fires discussed above and should therefore be incorporated in the final problem formulation.

To model the dynamic nature of forest fire monitoring some time dependent aspects should be introduced. Many such problem variants, solution approaches and applications derived from the OP are explored in the surveys by Vansteenwegen et al. [25] and Gunawan et al. [14]. These surveys cover results for established extensions to the original OP including (T)OP, (T)OP with time window (OPTW) and the time dependant OP (TDOP).

The newer variants: Generalized OP, the Stochastic OP, the Arc OP and the Multi-agent OP are also discussed, in order to determine the trend of future research. Algorithms for solving the mentioned problems are then compared for papers using standard or common problem instances. None of these are a particularly natural fit for our purpose, but considering stochastic behavior to travel times or scores may better model the uncertainty of the fires direction as a function of weather forecasts etc.

One of the best performing algorithms for the TOP according to Gunawan et al. [14] is due to Ke et al. [17], which combines many well known algorithms and concepts together with a so called Pareto dominance based rule to combine and alter solutions efficiently.

Zhang et al. [27] explore 5 algorithms for solving the VRP, but also consider many extensions, such as randomly moving threats and a 3-dimensional environment. Considerations of multiple objectives can be seen in Erdoğdu and Karabulut [13], where a Green VRP using electric vehicles is considered.

It is clear that a wide variety of routing problems have been considered and that there exist many algorithms to solve them or ones that should at least be adaptable. The above methods are only concerned with the nodes visited and the associated cost to reward trade-off, but rarely other aspects of the resulting paths.

One aspect of the routes that have received attention is captured in the aforementioned stochastic OPs. Panadero et al. [22] considers stochastic travel times for the TOP. Here a heuristic with an embedded simulation is used to evaluate how the variance in travel time affects the likelihood of violating the path length constraint. It does however not consider interactions between paths or how stochastic travel times affect it. An overlooked aspect therefore seems to be how agents in the TOP setting interact.

The most obvious concern is possible collisions due to path conflicts for different UAVs. This is

especially relevant when variations in speed, location etc., are taken into account. These issues can be side-stepped as in Khemakhem et al. [18], where the TOP is solved by creating clusters equal to the number of agents using variable neighbourhood descent and then solving an OP for each cluster. Here clusters are optimized using local search and each of the OPs are solved with a tabu search algorithm.

An important consideration when dealing with multiple agents in any system, is whether planning is done in a centralized or decentralized manner. Much of the existing literature on the TOP are assuming that planning can be done by a central authority. But as pointed out by Murray et al. [20] this assumption is not always realistic, particularly not for agents operating in an adversarial setting where the ability to communicate is restricted. In the case of forest fires, situations with limited communication can arise if for example cell towers are damaged by factors related to the fire, see Figure 1.2, or the infrastructure in the area is simply inadequate. The authors of the paper allow each agent to act independently seeking to maximize its own profit and call this the prize-collecting multi-agent orienteering problem (PCMOP).

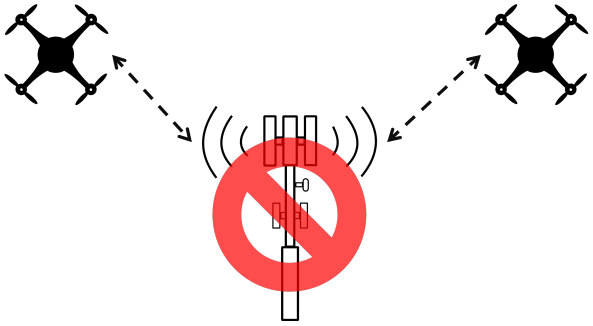


Figure 1.2: An illustration of how agents relying on communication via cell towers are vulnerable to these cell towers coming offline due to environmental impact during a forest fire.

Other studies of decentralized planning include Best and Hollinger [2, 3], where so called online orienteering for multiple agents are considered. Online in this context means the agents paths are updated during the mission. Decentralized refers to the agents only being able to modify their own routes, but not being able to inform others, which is justified by the assumption of limited communication between agents. The problem is then formulated as visiting points of interest given a range constraint, much like an OP with the an identical score for all nodes. This is mentioned as being suitable for e.g., surveillance and since many of the aspect considered in their problem formulation carries over to monitoring of forest fire, it seems a suitable initial formulation.

1.2 Statement of Intent

Based on the literature presented above, it seems that many variants of the VRP have been considered, but much of the nuance in real life applications, such as the interactions between the agents have been largely omitted, though this is to be expected for general problem formulations and solution methods. This project will therefore use some of these problem formulation as a starting point and incorporate aspects specific to forest fire monitoring and similar applications, in order to improve forest fire surveillance using UAVs that can adapt the route to environmental conditions. This is summarized in the following statment.

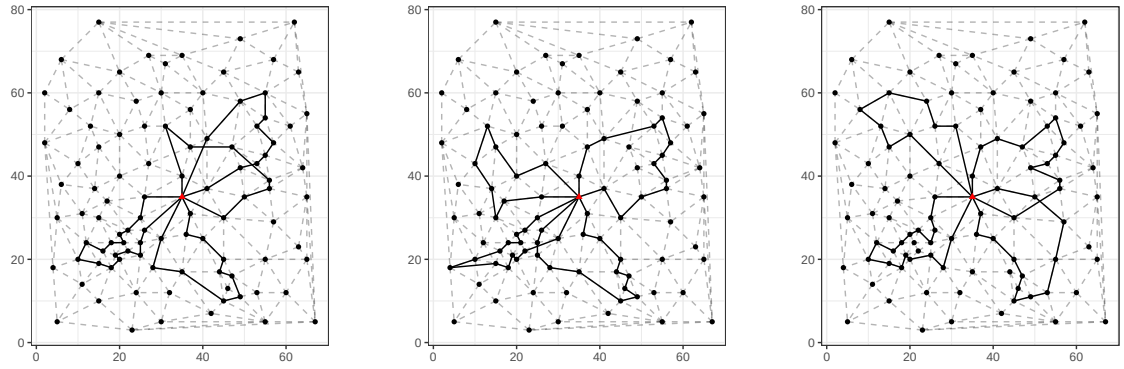
- *How can a solution to a team orienteering problem with environmental uncertainty be formulated, in such a way that each agent has the ability to update its own path, while ensuring that UAVs do not operate in the same area even with no ability to communicate with each other?*

2 Problem Formulation

The generalizations of the TSP mentioned in the introduction are collectively refered to as multi-level vehicle routing problems (MLVRP) by Chao [9]. As an example of a level in this sense is selecting the nodes to be visited in a TSP among a larger set of nodes, which is known as the Orienteering Problem (OP). The Team Orienteering Problem (TOP) is another added level in the MLVRP, where nodes and related edges must be uniquely distributed among a number of agents and performance is measured collectively.

Our proposal is to add an additional level to the TOP, where paths for different agents cannot intersect, in order to reflect safety and efficiency concerns in applications of the TOP. This aspect has received little explicit attention in literature, though it is often a by-product of having efficient routes. This is because the node visited by crossing the path of another agent is typically further from its path than the path of the agent it crossed. Examples of this can be seen in Figure B.356 and B.357 in Chao [9], that have been recreated in Figures 2.1b and 2.1c respectively, for examples of where resolving crossed paths could lead to visiting the same nodes using less range, similar to the idea behind the 2-Opt optimization algorithm for the TSP introduced by Croes [10]. There are however cases such as the route illustrated in Figure 355 from Chao [9], which is recreated in Figure 2.1a, where the effects of untangling the routes of different agents is less clear, but still has potential for improving efficiency.

Adding this constraint leads to a formulation where not only are the sets of nodes and edges distributed among the agents pairwise disjoint, but the paths for all agents also have no overlapping edges. This concept of separating the routes of different agents will be refered to as zoning and the resulting subsets of nodes, available to only one agent each, are called zones.



(a) A recreation of the TOP routing solution from illustrated in B355, Chao [9].

(b) A recreation of the TOP routing solution from illustrated in B356, Chao [9].

(c) A recreation of the TOP routing solution from illustrated in B357, Chao [9].

Figure 2.1: Recreations of TOP routing solutions from Chao [9].

An aspect of routing relevant for nearly all applications is how information gathered along a pre-planned route can be used to improve the remaining part of the route during a mission. In surveil-

lance applications this can be especially impactful, as one observation can potentially affect what area should have priority. Consider the monitoring of forest fires, where the development of the fire observed by the agent would influence which area would be most worthwhile to monitor given this new information or what areas can no longer be visited safely. In this way a feedback of information gathered based on the route completed so far can inform where the remaining route should go and so on for each time a new point of interest is visited or new information is otherwise gathered.

Also in rescue missions and other dynamic problems, a system for updating the remaining route based on new information could be useful, even if it is not gathered by the agent itself. This could be anything contributing to predicting the targets future behavior, such as satellite images, historical data about the targets previous behavior, etc. Lastly, restrictions to communication between agents will be considered, as this is an area where separating the agents path provides a significant safety improvement when changing the original route within the assigned zone, which is essential for responding to new information quickly and effectively. An overview of the routing aspects described above is given in Figure 2.2 and details will be explained in the next section.

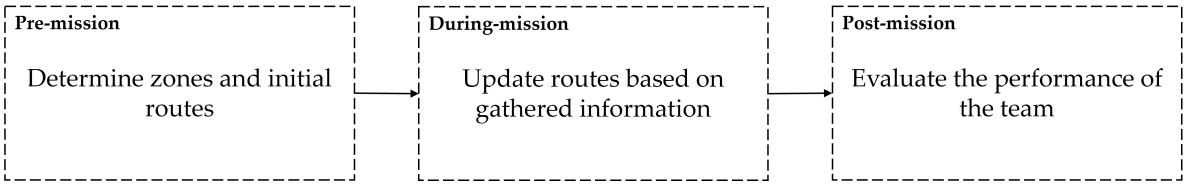


Figure 2.2: Timeline of the mission.

2.1 Mathematical Formulation

The problem will be described in the context of graph theory, wherefore the network structure will be presented first. A formal description of the problem will then follow, in order to make standard solution methods easily applicable.

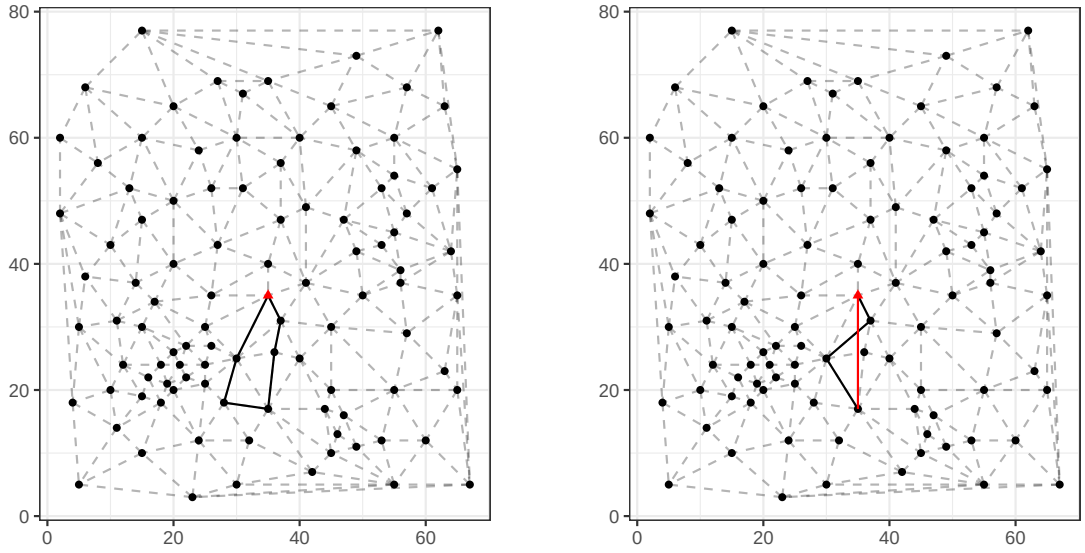
This representation is similar to many others found in VRP related literature, such as Chao et al. [8]. Consider the undirected graph $G = (N, E)$ defined by a set of nodes $N = \{1, \ldots, |N|\}$, where $|N|$ is the number of nodes in the graph and each node, i, is associated with a profit variable S_i. Node 1 represent the depot at which agents depart and arrive. Elements from the edge set E is denoted by $e_{(i,j)}$, which is defined as the edge connecting the nodes i and j, and is associated with the non-negative deterministic travel time $t_{e_{(i,j)}}$.

In our application, deciding how connected a graph should be is a trade-off between routing flexibility and the number of feasible solutions to be evaluated in order to determine optimality. A complete graph would be able to represent all feasible solutions to a classical TOP, but many of especially the long travel time edges would go unused in high quality solutions, as also seen in Figure 2.1. This is because in resource constrained routing applications, shorter edges means potentially fewer resources required to include a given node in the route.

For the purpose of avoiding crossing paths between agents, we will use Delaunay triangulation as the basis for all graphs seen in this report. A map with edges found by Delaney triangulation can be seen in Figure 2.3. This method avoids crossed edges and can improve routing efficiency as shown in [6, 16, 23]. During the procedure, of constructing the triangulation, when splitting a quadrangle into two triangles the shortest edge possible is used. This is again useful, since the cost of including a node is potentially lowered. Regarding the number of solutions, the number of edges is at most $2|N| - 2 - b$, where b is the number of nodes in the convex hull of the entire graph.

A route will be represented as an ordered list of the nodes visited and since the edges are uniquely determined by a pair of nodes this can represent all possible routes. An example of valid and invalid routes in the context of a graph with only Delaunay triangulation edges can be seen in Figure 2.3.

Note that the route in Figure 2.3b is invalid, even though it only crosses edges in its own route. Agents crossing their own route should not represent any risk, but should again not be part of high quality solutions. Additional edges can however be added within each zone after zoning, depending on the desired ratio between computation time and solution quality. Adding edges will however require some care, in order to guarantee that none of them cross edges from other zones.



(a) Illustration of a valid route.

(b) Illustration of an invalid route.

Figure 2.3: Route examples on a map with edges found by Delaunay triangulation.

2.1.1 Dynamic Aspects

In order to model how information gathered along a route can influence the optimality of the remaining route, score dependencies are introduced between nodes, that reflect how visiting one node could affect the score of other nodes. These are stored in a square information or adjustment matrix, A, where element (i, j) describes how observing something unexpected in node i influences the score of node j. Ideally this would be based on correlations seen in historical data or expert opinion of how the terrain is likely to act in a given situation. In a forest fire application this could be humidity of each area, placement of so called fire lines or other fire mitigation measures, and how this influences the spread pattern of the fire.

For the purpose of keeping this aspect of the project simple, these dependencies will be generated as a function of physical proximity. More precisely, element (i, j) of matrix A is generated by

$$(A)_{ij} = t_{e_{(i,j)}} \cdot r, \quad \text{where } r \sim unif(-1, 1). \tag{2.1}$$

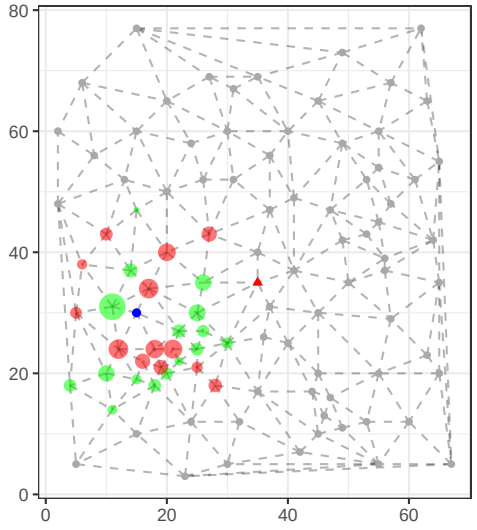
This introduces both positive and negative relations, in order to reflect the fact that areas can become either more desirable or dangerous to visit, when new information is considered.

Since the routes should only be updated when the assumptions they were based on have been violated, a way to determine this needs to be implemented for testing purposes, given that they are unknown pre-mission. This should again be problem specific and reflect the uncertainty in the scores or more accurately the factors they are based on. This could in many applications be weather forecasts and for forest fires specifically, wind direction, speed and humidity should influence what areas are visited. Other factors such as the distribution of people in the area are candidates for factors that should be dynamically adapted to.

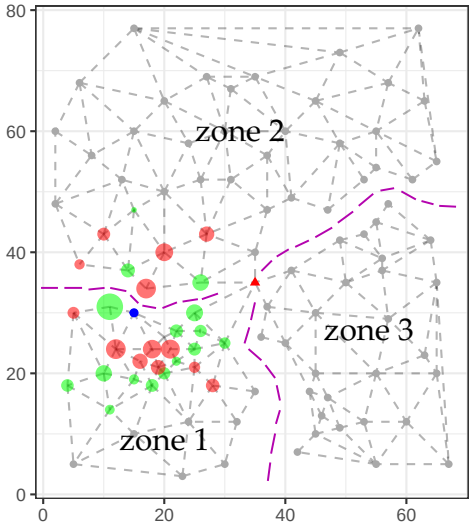
In our formalisation of this concept, each node will be assigned a probability, $0 \leq p_{u_i} \leq 0.5$, for something unexpected to happen, that warrant updating the score of related nodes as per the information matrix. Realizations of unexpected events are then drawn from a Bernoulli distribution using the probability p_{u_i}. The way a realization of an unexpected event affects the score of other nodes is visualized in Figure 2.4a. Given no information about the certainty of the score estimates, the probabilities p_{u_i} will simply be assumed uniform from the same distribution

$$p_{u_i} \sim unif(0, 0.5). \tag{2.2}$$

These dynamic aspects are straightforward to implement when updating existing routes, but less so when creating zones. The reason this matters for zoning is because allocating strongly related nodes to different zones significantly decreases our ability to optimally adapt the routes to new information. This issue is exacerbated when communication between agents is limited or impossible, as not even the agent with access to the related node can take the new information into account. This is illustrated in Figure 2.4b, where the score of 7 nodes in zone 2 should be updated, but since another agent serves these and may not even be close to them for a long time, this likely results in sub-optimal routes.



(a) Illustration of how an unexpected observation at the blue node is dispersed in the graph. The colored nodes are affected by the observation, green indicates positive effect whereas red indicates negative effect in terms of score. Node size indicate the size of the effect.



(b) Illustration of how an unexpected event in zone 1 should inform the score of nodes in zone 2, but may not be able to because of communication constraints between two agents.

Figure 2.4: Illustration of the effect and general pattern of dynamic score updating.

2.1.2 The Zoned Team Orienteering Problem

Using this graph structure and dynamic characteristics, the problem can now be formalized. The first step in expanding the TOP to a Zoned Team Orienteering Problem (ZTOP) is distributing the nodes between agents. The goal is therefore to create k disjoint subsets of N, $z_1, \ldots, z_k$, such that the cumulative score of the k OP solutions called routes, can be maximized.

The structure of the problem is based on a 2-stage stochastic programming problem. These problems are further explored in many papers, see Ahmed [1] for some intuition and the basis for our formulation. This refers to there being two optimization problems, where one of them depends on the stochastic behavior of the other. The relation to our problem is that the zones have to be decided before the routes, but the routes depend on the realization of unexpected events, that can not be

fully determined until after zoning is complete. Our first-stage choice is therefore to select what nodes, or equivalently edges, go to which zones and the second stage is to create a route from the nodes in each zone. We therefore need to use the routes we expect to get based on the scores we expect to get, in order to determine optimal zoning in a dynamic environment.

More formally, we want to maximize the expected score given the distribution of unexpected events $P = \{p_{u,1}, \ldots, p_{u,|N|}\}$, by allocation edges to zones z_v for $v = 1, \ldots, k$, given how this influences the optimization problem of selecting edges from each zone that then constitute a route. This is done using the binary decision variables x_{ij}, that determines if the edge from i to j is included in the given zone. The zoning optimization problem is

$$\max_{z_1,\ldots,z_k} \sum_{v=1}^{k} \mathbb{E}_P[Q_v(X_v, u)], \tag{2.3}$$

$$s.t.$$

$$z_v \subseteq DT(N), \tag{2.4}$$

$$z_v \cap z_w = \emptyset \text{ for } v \neq w, \tag{2.5}$$

$$i \to j \; \forall i, j \in N_v, \; \forall v = 1, \ldots, k, \tag{2.6}$$

where $X_v = \{x_{ij} \mid x_{ij} \in z_v\}$ are the edges allocated to zone v, $N_v = \{j \mid x_{ij} \in z_v\}$ are the nodes in zone v and $Q_v(X_v, u)$ is the realized score of the route in zone v given the realization u of unexpected events. DT is a function that returns the edges of a specific Delaunay triangulation of the input nodes. The objective function in Equation (2.3) adds all the scores gained by visiting nodes for each agent together, depending on the realizations of unexpected events, which again depends on the order in which each node is visited. To avoid agents crossing paths, Delaunay triangulation is used to obtain edges, which are then split among the zones, as seen in Equation (2.4). The connectivity constraint in Equation (2.6) then ensures that any node in a zone can be visited from any other node in the zone using only edges in that zone.

The second stage of the optimization problem, which concerns the routing, is actually multiple stages, since each edge added to the route changes the expected cumulative score and therefore potentially the remaining optimal route. Formulating it as a multi-stage linear problem would however complicate especially the length constraint as it partially determines the number of total steps in the formulation. So in an effort to increase readability and because this formulation will not be used directly in the solution approaches, this simplified 2-stage formulation has been chosen.

The edges that make up the routes are chosen by the decision variables $Y_v = y_1, \ldots, y_{|X_v|}$. These are binary variables that select the edges in X_v to use for the associated routes. This means that $Y_v X_v$ should be interpreted as the nodes from zone v used in route v. The routing optimization problem is

$$\mathbb{E}_P[Q_v(X_v, u)] := \max_{X_v} \sum_{j \in N_R} \mathbb{E}[S_j | Y_v X_v], \text{ where } N_R = \{j | x_{ij} \in Y_v X_v\} \tag{2.7}$$

$$s.t.$$

$$i \to j \quad \forall i, j \in N_R \tag{2.8}$$

$$\sum_{j \in \delta^+(i)} x_{ij} = \sum_{j \in \delta^-(i)} x_{ji}, \quad \forall i \in N_R \tag{2.9}$$

$$\sum_{x_{ij} \in Y_v X_v} x_{ij} l_{ij} \leq L, \tag{2.10}$$

The flow constraint in Equation (2.9) ensures that each node visited in the zone is left again and Equation (2.8) eliminates sub-tours by making sure that all visited nodes are connected using only edges in the zone. Lastly, Equation (2.10) ensures that the UAVs range constraint is respected.

The relation between the order nodes are visited in and the score of other nodes is dependent on unexpected events. This is formalized by the conditional expectation of the score for each node as

seen here.

$$\mathbb{E}[S_j|Y_vX_v] = \mathbb{E}[S_j] + \sum_{i \mid (x_{ij} \in Y_vX_v)} \left(A_{ij}u_i - A_{ij}p_{u_i}\right) \tag{2.11}$$

$$= \mathbb{E}[S_j] + \sum_{i \mid (x_{ij} \in Y_vX_v)} A_{ij}\left(u_i - p_{u_i}\right), \tag{2.12}$$

This means that the pre-mission score for node j is adjusted after an edge is added to the route by removing the uncertainty from the last term in Equation (2.11) and adding the effect of visiting node i, according to their dependence given by A_{ji}, as seen in the second to last term in the sum. The rewriting in Equation (2.12) shows that $u_i - p_{u_i}$ can be interpreted as how unlikely the event was to occur and therefore the scale of its impact on the score. Note that the sum in (2.11) is only over nodes in its own route, meaning no information shared from other agents is used.

This formulation has a few important advantages and disadvantages when compared to the standard TOP. Unlike the TOP, each OP becomes decoupled when the zones have been decided. The ZTOP therefore simplifies to an OP for each agent that can be solved independently, but requires an upfront investment in an efficient distribution of nodes. Given the typically greater than linear complexity of solution methods for orienteering problems with respect to the number of nodes, solving for a subset of the nodes at a time will likely reduce total run time, as compared to a more traditional TOP where more nodes are considered by a given agent. The of drawback zoning is then the complex problem of assigning nodes to zones effectively.

2.1.3 Problem Instances

The graphs, that when combined with parameters, number of agents and range, will be refered to as instances are described in Chao [9] and used in the TOP context seen in Chao et al. [8]. We will focus on the variations of the one called p7, since it has some desirable qualities. Relevant for the ZTOP is that it has at least as many nodes connected to the source and sink as the number of agents. Additionally, it has a centrally located and shared source and sink allowing for zoning flexibility. It also has higher scoring nodes far from the base and varying score density, among other characteristics, that increases difficulty and therefore hopefully allows us to test different aspects of the zoning solutions quality.

3 Solution Methods

A number of straight forward general approaches comes to mind when thinking about a solution to the ZTOP. One approach would be to partition the area into distinct zones and then construct routes within the boundary of these zones. This approach can be called "zone first route second". Another approach would be to construct routes sequentially, with each route staying clear of the previously constructed route. Thus the approach consists of first constructing the routes and then making the zones fit around the constructed routes. This approach can be called "route first zone second".

The main priority of the two approaches are what is done in the first step, either zoning or routing. Strengths of the "zone first route second" approach are flexibility in the zone construction and the computational advantage of being able to construct routes in parallel, as the zones are constructed before the routes. On the other hand the strengths of the "route first zone second" approach is that the flexibility is in the route construction, so the potential for constructing optimal routes are at least as good the other approach.

The difference in priority between the solution approaches are illustrated in Figure 3.1. As can also be seen on the figure, is a third approach called the "hybrid approach", that lies somewhere between prioritization of zoning and routing.

Another consideration to take into account is the dynamic aspect i.e. the effect of unexpected events. Because one of the main advantages of having established zones for each agent is that an agent is able to navigate within its zone at its own discretion, the anticipation of route updates should be incorporated into the derivation of the zones.

The aim of this chapter is to present a "zone first route second" approach as well as a "hybrid approach" trying to capture the strengths of both approaches. After discussing zoning approaches the chapter will continue by outlining routing algorithms for both pre-mission construction of routes and during-mission route updating.

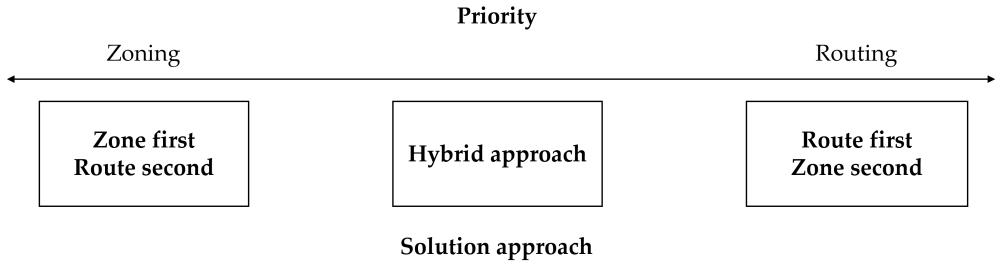
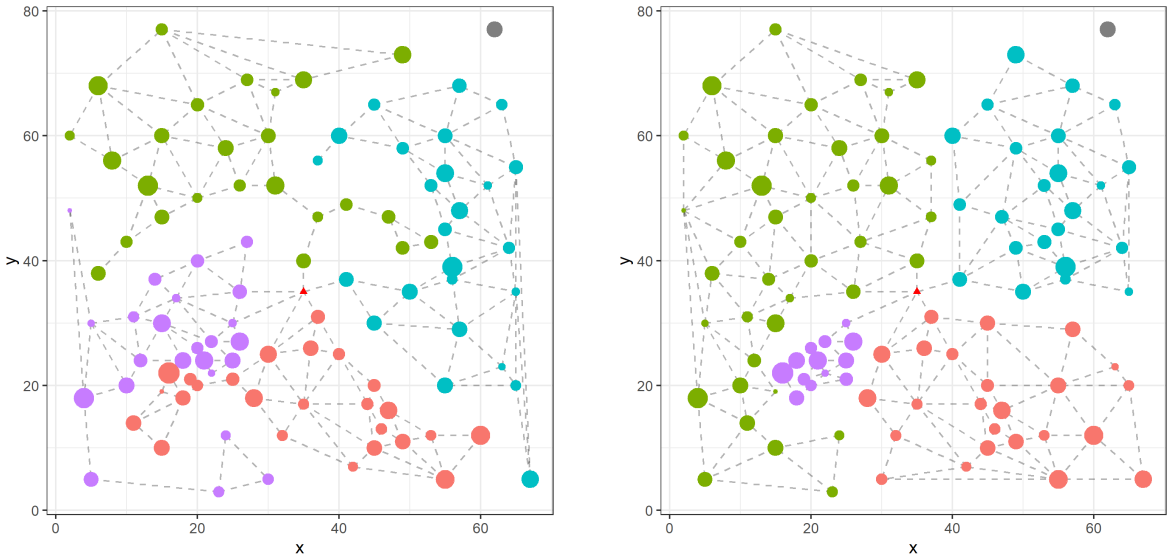


Figure 3.1: The relationship between different solution approaches and their prioritization between zones and routes.



(a) The zones from the initial greedy assignment of nodes before conducting the local search.

(b) Zones obtained after conducting the local search.

Figure 3.2: Examples of zones made with the heuristic-based approach.

3.1 Zoning Approaches

This subsection will first present a "zone first route second" approach which will be called the heuristic-based approach and then a routing-based "hybrid" approach will be presented afterwards.

3.1.1 Heuristic-based Approaches

This solution approach is inspired by the clustering method used by Khemakhem et al. [18], as a step in solving the SVRP. As mentioned earlier this approach is refered to as zone first route second, since zones or clusters, as they are called here, are decided before any routes have been considered. The idea is that all the aspects that influence routing can be accounted for when clustering and thereby not hinder routing performance.

The aspects of the graph accounted for by the clustering method in Khemakhem et al. [18], is the score and edge weight refered to as distance. This is used to evaluate each clusters compactness, measured as the average distance between nodes, and the clusters total score. Each clusters suitability is evaluated by the ratio between its compactness and total score, such that agents can visit the most possible nodes in the zones with highest score. More formally, the following objective is minimized

$$\xi(z_v) = \frac{\sum\limits_{i,j \in z_v} l_{ij}}{|z_v|(|z_v| - 1) \sum\limits_{i \in z_v} S_i} \tag{3.1}$$

where z_v is the vth zone and $|z_v|$ is its cardinality. This objective is first used to greedily assign nodes to zones, that initially only include the base node, by evaluating for which zone this objective improves most. An example of an initial greedy assignment of nodes can be seen in Figure 3.2a.

This is then followed by a local search procedure that uses the same objective to move nodes between zones. Khemakhem et al. [18] chooses between two operators, based on each clusters cardinality, in order to ensure appropriate cluster sizes. We only use the so called insertion move, where a node is

removed from one cluster and inserted into another. This is due to the parameter tuning required for each map to determine how much the cardinality of the cluster are allowed to differ, such that the local search results in appropriate cluster sizes. An example of zones constructed using this approach can be seen in Figure 3.2b.

In order to reflect the dynamic introduced by the ZTOP, we wish to formulate an objective function used in the same framework as above, that incorporates the dynamic aspects of the problem. The information matrix will be used to determine which nodes should be in the same zone, according to how much visiting one of them can influence the optimality of including the other in the route.

We therefore wish to maximize total dependency between nodes in each zone, to optimize route updating flexibility. This means maximizing the following objective function for each zone

$$\zeta(z_v) = \sum_{i,j \in z_v} A_{ij}. \tag{3.2}$$

Using this to assign nodes to zones, should allow routes in these zones to be effectively updated, with regards to the dynamics specified. It also emphasises assigning all available nodes to a zone, as adding nodes will always improve the objective function value. Lastly, large zone boundaries are encouraged, since A_{ij} is a function of distance and these are therefore more likely to be in the same zone. This often also results in similarly sized zones, likely due to the way A_{ij} is generated.

This objective function can be used either as an extension of the previous, with a multi-objective function or alone as its own method. As a starting point it will be considered as its own solution method, for which advantages and disadvantages can be evaluated, in order to determine its suitability before considering more advanced interactions between objectives.

3.1.2 Routing-based Approach

The purpose of this section is to present a "hybrid" approach which will be called the routing-based approach. As mentioned in this chapters introduction, the aim here is to have more emphasis on the routing aspect, as compared to the previous approaches. The routing-based solution approach can be described by the following 3 steps.

1. **Route construction**

   Construct multiple routes based on a greedy randomized route generation algorithm, incorporating the expected effects of unexpected events. The route construction is set up to obtain a balance between exploration and exploitation of the map.

2. **Clustering**

   Group the constructed routes into clusters based on the similarity between routes. These clusters will then consist of a set of routes that in turn consists of a sequence of nodes. The unique set of nodes visited by routes in the cluster will then form the basis of the zone, however these sets can not be guaranteed to be disjoint.

3. **Resolving disputes**

   The last step is to transform the cluster node sets such that the nodes in each set are connected and the sets are pairwise disjoint, except for the source node which is included in all zones.

The intuition behind this approach is to prioritize routing and zoning together by using an unsupervised learning model to create the zones based on already constructed routes. In the following 3 parts, each step will be explained in more detail, while pseudocode for the approach is found in Algorithm 3. Further explanation for the terminology in the algorithm pseudocode will be given in the following section.

Algorithm 1: Greedy Randomized Route Generation (GRRG)

Input: instance, distances, L, top_percentile

```
route <- c(1)                                          # Initialize route with the source node
current_node <- 1                                      # Keep track of the location of the agent
L_remaining <- L
route_concluded <- FALSE
while (!route_concluded) do
    # Obtain SDR values and names of candidate nodes, SDR are obtained using Algorithm 2
    sdr <- get_SDR(current_node, L_remaining, expected_score, distances, top_percentile)
    candidates <- names(sdr)
    if (length(candidates) > 1) then
        # If there is more than one candidate the next node is sampled
        next_node <- sample(candidates, size = 1, prob = sdr)  # Sampling according to SDR value
    else if (length(candidates) == 1) then
        # If there is only one feasible candidate, that node is selected
        next_node <- candidates[1]                     # Pick the first entry in the vector
    else
        # If there are no feasible candidates, go to sink and conclude the route
        next_node <- 1
        route_concluded <- TRUE                        # Exit the while loop in the next iteration
    path_to_next <- sp(current_node, next_node)        # Find the shortest path to the next node
    route <- append(route, path_to_next)               # Add the path to the route
    L_remaining <- L_remaining - distances[current_node, next_node]
    current_node <- next_node                          # Update the location of the agent
return route
```

Route construction

The algorithm for route constructing is outlined in Algorithm 1. As input the algorithm takes an "instance" which refers to the graph comprised of nodes with associated scores and edges with associated weights. It works by iteratively selecting the next node in the route while ensuring that the agent will have enough range to return to the source node.

The criteria for selecting the next node is based on the score to distance ratio (SDR), calculated as the score of the node divided by the distance needed to reach the node. As an example, let us assume that we are currently in node i. The SDR of node j is then

$$SDR(i,j) = \frac{S_j}{d(i,j)}, \tag{3.3}$$

where $d(i,j)$ refers to the distance needed to travel from node i to j. This measure is similar to others used for solving routing problems, such as for the TOP in Chao et al. [8]. Given that the edges in the graph come from a Delaunay triangulation, each node in the graph is only connected to a limited number of neighbors. To increase the possible neighbors to select among in each step, the algorithm is allowed to select between all nodes in the graph. If a non-neighboring node is selected the shortest path, found using Dijkstra's algorithm [12], to that node will also be added to the route.

In the pseudocode the shortest path is calculated using the "sp" function and the length of shortest path between all nodes is stored in the "distances" matrix. Additionally when dealing with non-neighboring nodes the SDR is generalized by also including score collected along the shortest path

Algorithm 2: get_SDR

Input: current_node, L_remaining, score, distance, top_percentile

```
# List of shortest from current node to all_nodes in the zone
paths <- sp(current_node, all_nodes)
s <- c()
for (i in paths) do
    s[i] <- score[unique(i)]                          # Score collected from each of the paths
d <- distance[current_node, all_nodes]                    # Length of the paths to other nodes
feasible <- d + dst[,1] <= L_remaining    # Nodes that can be reached within the remaining range
sdr <- s/d * feasible                                                  # SDR of feasible nodes
candidates <- sdr[sdr > 0]                          # Candidate nodes are positive SDR nodes
candidates <- sort(sdr, decreasing = TRUE)                          # Sort the vector by SDR
# Find the top x percentile of feasible candidates and return the least integer greater than or
  equal to this value
candidates <- candidates[1:ceiling(length(candidates) * top_percentile)]
return candidates
```

to the selected node

$$SDR_g(i,j) = \frac{\sum_{n \in sp(i,j)} S_n}{d(i,j)}. \tag{3.4}$$

To explore the search space the node selection uses a randomized approach. This means that at each step the generalized SDR is calculated for all nodes and then the next node is sampled among the top x percentile weighted by their relative SDR. Here x acts as a tuning parameter that determines how much we will compromise the exploitation at each step, in favor of exploration. Where $x = 0$, i.e. a top percentile of 0%, is defined as always choosing the node with highest SDR. On the other hand with $x = 1$, i.e. a top percentile of 100%, is defined as selecting the next node randomly between all candidates.

To accurately asses the appropriate value for x, parameter tuning should be performed. In general we found that a lower top percentile would give higher route scores, up to a certain point. This can be seen in more detail in Figure 3.4. However route score is not the only concern as its also desired to explore the search space. Given this trade off the value set will therefore be $x = .5$ for the remainder of this project.

Having established the route construction procedure the desired number of routes can now be constructed using Algorithm 1. Note that the number of routes, nr, to construct in this first step is in itself a tuning parameter of the algorithm. Based on experimentation we determine $nr = 1000$ to be sufficient for the specific test instance used in this project. In the clustering step, this approach will need to calculate the similarity measure $\frac{nr^2}{2} - nr$ times, assuming a symmetric similarity measure, wherefore much higher values of nr are not practical. For illustrative purposes a continuing example will be shown at each of the three steps, using $nr = 100$ for simplicity. The first example of route construction can be seen in Figure 3.3.

Clustering

The idea behind the clustering step is that zones should be based on areas where routes are clustered together. In order to quantify the spatial separation between routes we need a distance measure. Bockholt and Zweig [4] introduces a number of measures that can be used to compute similarity between paths in a graph. One of these is the matched average distance (MAD) that is defined

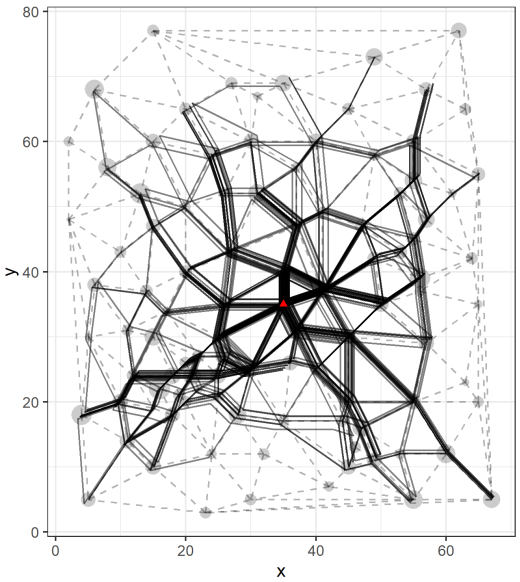


Figure 3.3: An example of 100 routes constructed by Algorithm 1. To better illustrate the route density in a given area, noise have been added to the x and y coordinates for each route. As expected there is high density around the source node, but as an example also the path to the node in the bottom right corner is frequently used.

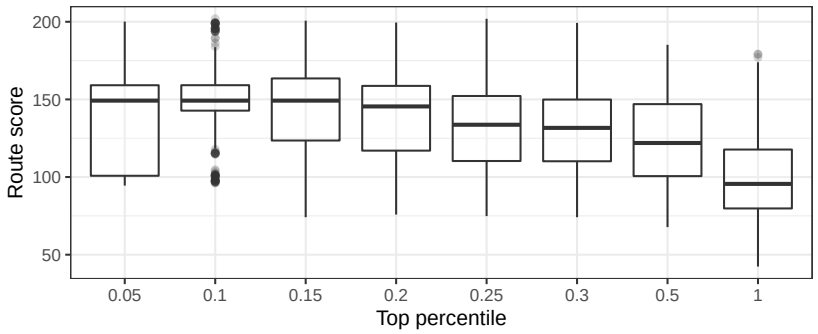


Figure 3.4: Effect of top percentile of nodes considered, for choosing next node in the initial routes, on cumulative route score.

between two routes P and Q as

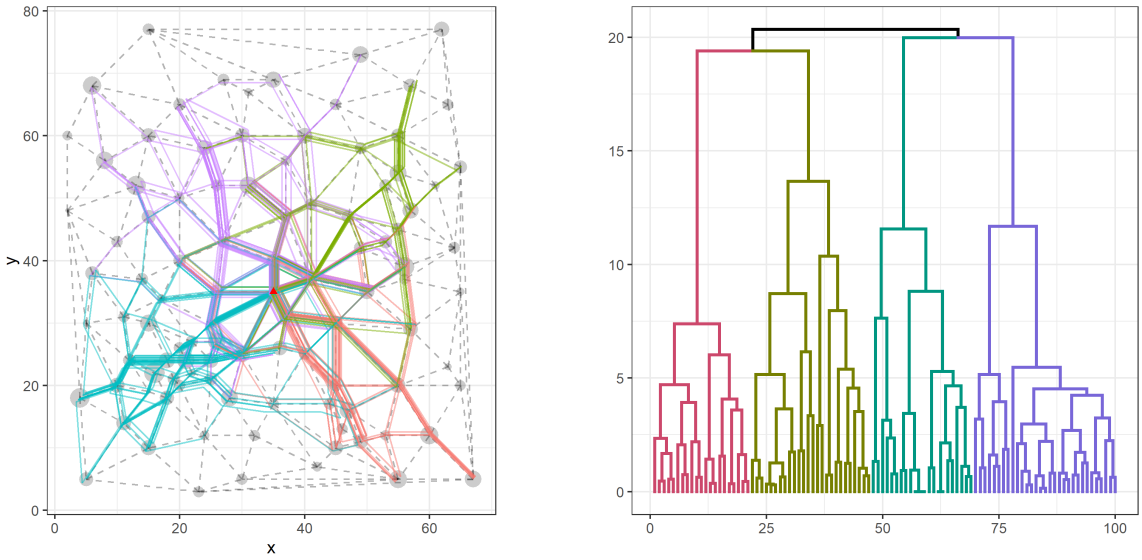
$$\text{MAD}(P,Q) = \frac{1}{|P|}\sum_{i=1}^{|P|} d\left(P_i, Q_{G(i)}\right) \qquad G(i) = j \text{ s.t. } j = \arg\min_j d\left(P_i, Q_j\right), \tag{3.5}$$

where $|P| \geq |Q|$ and $d(i,j)$ denotes the length of Dijkstra's shortest path between node i and j. In other words; for each node in the longer route we match this node with the closest node in the shorter route and then we take the mean over these distances.

Having established the MAD metric we use this together with hierarchical clustering to find the desired number of clusters. For this purpose we utilize an agglomerative hierarchical clustering algorithm, see James et al. [15, p. 526]. To determine distance between sets of routes we utilize complete-linkage, such that the distance between two sets of routes, U and W, is given by

$$\max\left\{\text{MAD}(P,Q) \mid P \in U,\ Q \in W\right\}. \tag{3.6}$$

Where single-linkage clustering can tend to produce clusters that are more spread out, for this application it is desired to have clusters that are more dense to achieve a high degree of similarity between routes in a cluster.



(a) Routes are colored according to cluster assignment. As can be seen the clusters are somewhat separated but there is still overlap between zones.

(b) The dendogram obtained from the hierachical clustering with branches colored according to doing a split at $k = 4$ zones.

Figure 3.5: The result of performing hierarchical clustering with $k = 4$ on the 100 routes from Figure 3.3.

To create zones from the clustered routes the next step is to transform the routes into sets of distinct nodes visited by routes in the clusters. Denote the set of routes in cluster v by C_v, the distinct nodes visited can then be derived as

$$Z_v^* = \left\{ i \mid i \in \bigcup_{j=1}^{|C_v|} R_j \right\}, \tag{3.7}$$

where R_j is a route contained in C_v. The sets Z_v^* for $v = 1, \ldots, k$ can not be guaranteed to be pairwise disjoint, as routes from two different cluster could utilize some of the same nodes. How we solve these node disputes will be the topic of the next section.

Resolving disputes

For the nodes that belong to multiple zones, it needs to be determined which single zone they should belong to, in order to make the zones pairwise disjoint. The disputed nodes are defined as

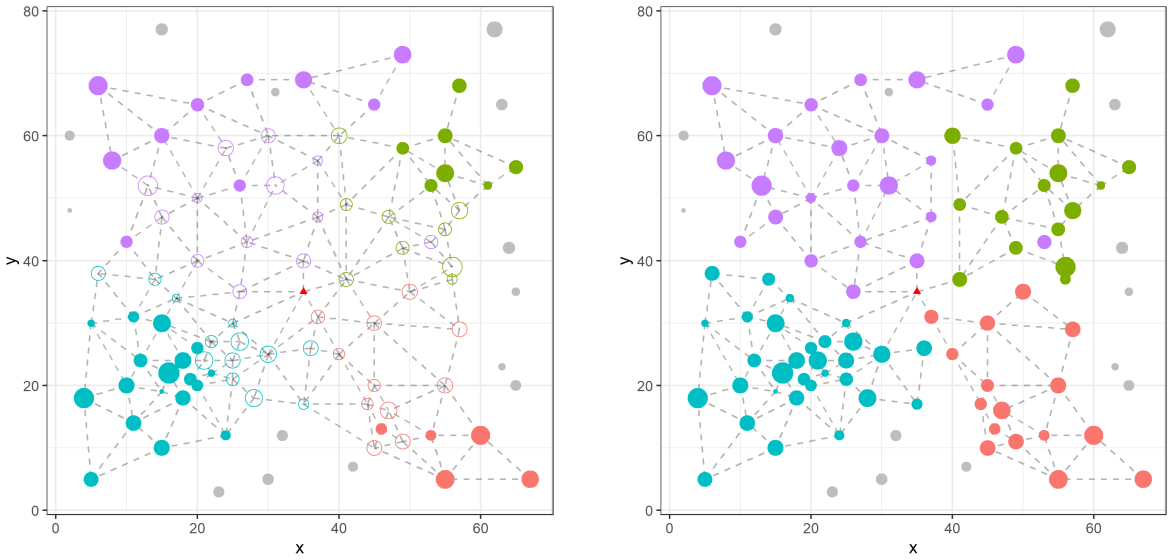
$$DN = \bigcup_{v,w \in 1,\ldots,k} (Z_v^* \cap Z_w^*), \tag{3.8}$$

in other words the set of nodes that lies in the intersection between any two zones.

To determine which zone a disputed node should belong to, we introduce a criteria based on most frequent usage. The normalized usage of a node by a cluster can be written as

$$\mathrm{U}_n(C, x) = \frac{1}{|C|} \sum_{R \in C} \mathbb{1}_{x \in R}, \tag{3.9}$$

that is, the fraction of routes in cluster C that includes the node x. Disputes are then resolved by assigning nodes in DN, from Equation (3.8), to the zone with the greatest normalized usage of that node.



(a) Illustration of how the zones look before node disputes have been resolved. Undisputed nodes are illustrated by a solid colored circle whereas disputed nodes are shown as colored circle with no fill. The disputed nodes are colored according to which cluster have most frequent usage of the node.

(b) Here the node disputes have been resolved by distributing the disputed nodes to the cluster that have the most frequent usage. Note that the zones are now disjoint but a connectivity issue have been introduced, as there is a purple node that is not connected to any other purple nodes.

Figure 3.6: Resolving node disputes from the clusters generated in Figure 3.5.

The reason for expressing a clusters node usage as a fraction as opposed to just the number of routes that use a certain node, is that clusters can consist of a different number of routes. If the disputed node were distributed according to absolute usage, already large zones would gain an advantage and thus grow even bigger which would not be desirable.

An illustration of this procedure can be seen in Figure 3.6, but here it is also seen that the approach can introduce connectivity issues within the zones. In this case one of the purple nodes is stuck in the green zone, not connected to other purple nodes. To mitigate this issue a post-processing step is introduced after resolving the node disputes. The objective being that every node in a zone should be connected to all other nodes in the zone, as described in (2.8). It works by first identifying the nodes that are not connected to the main part of their zone, defined as the part of the zone that is connected to the source, then storing these nodes. These nodes will be called “available”. Then we iterate through the available nodes and check if the node can be added to a zone without introducing connectivity issues in the zone. If the node can be added to multiple zones, the most compact zone will be selected.

3.2 Routing

This section will describe how routes are constructed, used and updated throughout the mission. The structure of this section will follow the chronological order of a mission, see Figure 3.7, such that the so called pre-mission considerations and decisions are described first. This is followed by the during-mission stage where the update schemes and associated decision making is described. The section then concludes by presenting post-mission analysis methods and their relevance.

After clustering has been completed and final zones decided, routes can be constructed for each cluster, irrespective of the routes used in routing-based clustering. These routes are therefore made in the exact same way regardless of the clustering method. These routes are refered to as starting routes and will be described in the next subsection.

Algorithm 3: Routing-based clustering

Input: instance, L, k, num_routes, top_percentile

```
# First step is to construct the desired number of routes to base the clustering on
initial_routes <- list()
for (i in 1:num_routes) do
    # Initial routes are made using the GRRG function and added as the i'th element of the list
    initial_routes[[i]] <- GRRG(instance, L, top_percentile)   # GRRG is described in Algorithm 1
# The second step is to compute the distance matrix and perform the clustering, note that only
  the lower triangle of the distance matrix is needed as MAD(x, y) = MAD(y, x)
combinations <- C(n = num_routes, k = 2)   # The number of 2-combinations of initial routes
dist_matrix <- matrix(nrow = num_routes, ncol = num_routes)
for (id1, id2 in combinations) do
    dist_matrix[id1, id2] <- MAD(initial_routes[[id1]], initial_routes[[id2]])   # Equation (3.5)
# Obtain clusters from hierarchical clustering
route_clusters <- hclust(dist_matrix, method = "complete")
# The third step is to resolve any node disputes between route clusters
for i in 1:k do
    Assign source node to zone i
for i in nodes do
    if node i is in multiple clusters then
        Assign node i to cluster with highest utilization
    else if node i is in a single cluster then
        Assign node i to the associated cluster
    else
        Leave the node unassigned
# Correct any connectivity issues in the zones
for all zones do
    if the zone is not connected then
        Set the nodes not connected to the source as available
while there are available nodes left do
    for all available nodes do
        for all zones do
            if the zone is still connected after adding the node then
                Store the available node and zone as a candidate assignment
                candidate objective = mean shortest path between all pairs of nodes in the zone
    Select the candidate assignment with lowest candidate objective
```

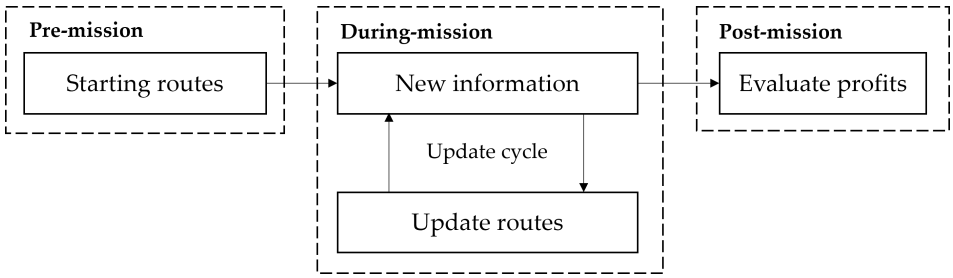


Figure 3.7: The stages and progress of a mission from a routing perspective.

3.2.1 Pre-mission

The first set of routes, one for each zone, are constructed as described in Algorithm 4. The intuition is that the shortest paths to all other nodes in the zone, called candidates, are considered by evaluating their suitability, similar to how it was done for the initial routes in algorithm 1.

The route is initialised by including the source and sink nodes. Each point is then experimentally inserted in the route before the sink and the distance used in the SDR measure is therefore calculated as the distance between the current last node before the sink to the candidate node, plus the distance between the candidate node and the sink. This therefore reflects the total length added to the route, in order to insert the candidate node before returning to the base. The score in the SDR calculation uses all the nodes in the shortest path, in order to reflect the change in the routes SDR after a given nodes inclusion in the route. This SDR calculation can therefore be undefined, when we would have visited a candidate on the way to the sink from the current node. Among nodes with undefined SDR we will only consider the node as a candidate if the score is positive and then it will be the highest priority.

This adding of nodes before the sink is then repeated until it is no longer feasible according to the range constraint in Equation (2.10). Examples of some routes resulting from this procedure can be seen in Figure 3.8. When there is not enough range remaining to add the highest SDR node, the other candidates are considered in descending order of their SDR. This is then added to the route in the same way as before, repeated until no further nodes can be added. When all nodes have been considered for insertion before the sink and none of them can respect the range constraint, the route is returned without any additional nodes added.

These starting routes can be used for the during-mission step, but a route improvement step will be done before this. Since the during-mission stage is time sensitive, as the decisions should be made on the fly, these improvements will be made in the less time sensitive pre-mission stage. This method of multiple optimizations of route concluding with a feasible solution, allows for flexibility in choosing a trade-off between solution quality and computation time.

These improvements are described in Algorithm 5. The first part of the improvement algorithm, seen in lines 5-17, use the same SDR criteria as the starting routes to find where inserting a node is most beneficial. More precisely, the insertion of all nodes at all entries in the route are considered and their SDR is calculated in a similar way to Algorithm 4. The best candidate is then chosen in lines 19-20 and 23. The resulting routes from considering shortest paths are updated accordingly in lines 24-27. The algorithm then terminates when there is not enough range to insert additional relevant nodes cf. lines 28-29 and 21-22.

It would seem obvious to combine the insertion procedure described above with remove or swap operators commonly used in local search optimization for VRPs. This will not be explored further in this project in favour of focusing on clustering performance and comparisons of the different methods proposed. It is however a straight forward extension of the methods presented here and the influence of improved routing performance may also influence the effectiveness of zoning.

Other post-processing optimization methods may also provide benefits in uncovering the effect of a clustering approach on applications with large time-budgets. Our experience is however that these improve solution quality very little given the required run time. Exploring such methods is therefore left as potential extensions that will be further discussed in Chapter 5.

Algorithm 4: Generate starting routes

Input: instance, zones, zone_id, L

```
L_remaining = L
candidates = zones[[zone_id]]                    # Possible candidates are other nodes in the zone
route = c(1,1)                                   # Initialize route with first and last node to visit
last_in_current = route[length(route)-1]
while L_remaining > 0 do
    # Find the change in SDR when including a candidate in the route
    for (i in 1:length(candidates)) do
        # Initialize temporary route to evaluate and add candidate i as the next to last node
        route_temp = route route_temp = append(route_temp, candidates[i], after =
          length(route_temp)-1)
        # The difference in distance of adding candidate i
        d[i] = distances(last_in_current, candidates[i]) + distances(candidates[i], 1) -
          distances(last_in_current, 1)
        path_to_candidate = sp(last_in_current, candidates[i])          # Path to candidate i
        s[i] = sum(score[unique(path_to_candidate)])  # Score gathered on the path to candidate
          i
        if (d[i] ≤ 0 & s[i] > 0) then
            SDR[i] = Inf          # Prioritize nodes that result in score while not adding distance
        else
            SDR[i] = s[i]/d[i]
    next_node = which.max(SDR)                   # The next node is selected to maximize SDR
    path_to_next = sp(last_in_current, next_node)
    for (node in path_to_next) do
        s_total = s_total + score[node]       # Add collected scores on the path to the next node
        score[node] = 0                           # Reset score as it can only be collected ones
    # We check whether the next node is within range
    while (distances(last_in_current, next_node) + distances(next_node, 1) -
      distances(last_in_current, 1)) > L_remaining) do
        # If the next node is out of range, the SDR is reset and the next best is checked
        SDR[next_node] = 0
        next_node = which.max(SDR)
        if (SDR[next_node] == 0) then
            # If all nodes have been checked and none are within range the route is concluded
              after adding the return path
            return_path = sp(last_in_current, 1)
            route = append(route, return_path, after = length(route)-1)
            return route, L, L_remaining
    # The next node is in range so it is added to the route
    route <- append(route, path_to_next, after = length(route)-1)
    L_remaining = L - route_length(route)
```

Algorithm 5: Improve starting routes

Input: route, L_remaining, L, zones, zone_id

```
# Save range for update routing
L_remaining = L_remaining/2
score[route] = 0                                   # Reset score for already visited nodes
# Possible candidates are nodes in the zone that are not in the route
nodes_in_zone = zones[[zone_id]]
candidates = nodes_in_zone[!nodes_in_zone %in% route]
while (L_remaining > 0) do
    for (n in 1:(length(route) - 1)) do
        for (i in 1:(length(candidates))) do
            # Initialize temporary path to evaluate and add candidate i as the n+1'th node
            route_temp = route route_temp = append(route_temp, candidates[i], after = n)
            # The difference in distance of adding candidate i as n+1'th node
            d[[n]][i] = dist(route_temp[n], candidates[i]) + dist(candidates[i],
              route_temp[n+2]) - dist(route_temp[n], route_temp[n+2])
            # Additional nodes visited to and from candidate i
            nodes_visited = unique(c(sp(route[n], candidates[i]), sp(candidates[i],
              route[n+1])))
            s[[n]][i] = sum(score[nodes_visited])      # Score on the path to and from candidate i
            if (d[[n]][i] > L_remaining) then
                SDR[[n]][i] = 0          # If the candidate is not within range, its SDR is set to 0
            else if (d[i] ≤ 0 & s[i] > 0) then
                SDR[[n]][i] = Inf          # Prioritize nodes that result in score while not adding
                  distance
            else
                SDR[[n]][i] = (s[[n]][i])/(d[[n]][i])
    for (nr in 1:(length(route) - 1)) do
        # Find the best candidate for each position in the route
        new_node[nr] = which.max(SDR[[nr]])
        value[nr] = max(SDR[[nr]])
    if (max(value) == 0) then
        return route                  # The route is returned when no candidate have positive SDR
    new_node_placement = which.max(value)      # At which place in the route should the new
      node be place
    # Find the shortest path to and from the new node
    path_to = sp(route[new_node_placement],
      candidates[new_node[new_node_placement]])
    path_back = sp(candidates[new_node[new_node_placement]],
      route[new_node_placement + 1])
                                        # Add the paths to the route at the new node placement
    route = append(route, path_to, after = new_node_placement)
    route = append(route, path_back, after = (new_node_placement + length(path_to)))
    L_remaining = L - route_length(route)
    score[c(path_to, path_back)] = 0                   # Reset scores for the nodes visited
```

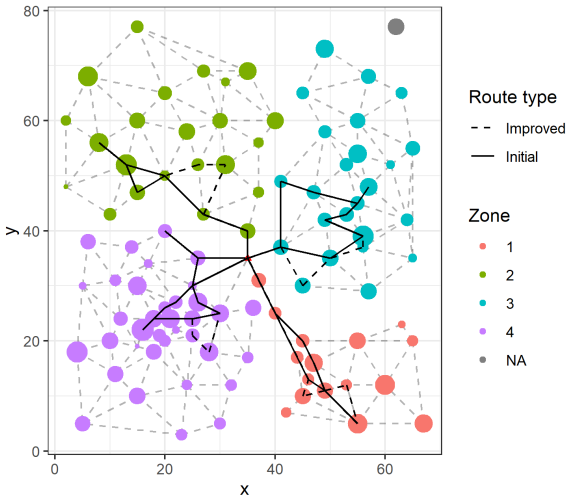


Figure 3.8: Example of starting routes, before and after the improvement step.

3.2.2 During-mission

Using the routes previously described, we can now move on to the during-mission phase, where methods for reacting to the score changing dynamics seen in chapter 2 will be introduced.

This is done using the route updating procedure seen in Algorithm 6. This algorithm takes any feasible route as input and can therefore use starting routes, improved routes or routes from any other OP solver as input. The intuition is that after visiting each node we evaluate if the planned next node in the input route still has the highest SDR, based on the updated scores. If not, it is replaced with the one that does and this is then repeated until the agent reaches the base again. This is illustrated with examples of a zoned instance in Figure 3.9.

The updating algorithm considers a few more aspects of the routes and the details will be given here. Whether an unexpected event occurs at a node is stored in $unexpected[node\_id]$, that is discovered as each node is visited, and is used in line 14 to update the scores of related nodes. The SDR calculation used in line 16, that is further explained in Algorithm 7, also considers nodes outside the zone, so that we can evaluate clustering performance post-mission.

Algorithm 7 works by considering the trade-off in travel time and the effect it has on SDR, when replacing the node we planned to visit next with each of the candidates respectively. The candidates here are refered to as neighbors in line 2-3, since they have to be directly connected to the node the agent currently occupies. It does however consider the shortest path back to the remainder of the planned route described in lines 4-7, as to not restrict updating flexibility too much. The neighbors that can then be reached are considered and chosen among according to their SDR in the usual way.

The node selected by Algorithm 7, for use in the next part of Algorithm 6 itself, is seen in line 20. Here we have chosen that if the best candidate is planned to be visited later in the route, based on the initial route, it is ignored in favour of the originally planned route. This is because it allows us to gather more information before visiting it and thereby allowing us to make a better informed decision. Also, assuming the initial route was efficiently made, this should also results in a lower range cost to visit the same node.

Since the measure used to determine which node to visit is the SDR, the range only has an upper limit in order to ensure feasibility. Therefore a specific range usage is not optimized for and can theoretically vary greatly. Based on preliminary testing, updating the route does not seem to lower

its length and making sure that this potential extra range is used therefore has a low priority. The problem instead seems to be using the available extra range too soon and not having enough range available to update later parts of the route, while still being able to complete the rest of the starting route.

There may therefore be an optimal value for range headroom when constructing the starting route instead of allowing it to use all the range. This is the reason for using half of the remaining range for improving the route in Algorithm 5. A method for guaranteeing a level of flexibility when updating would however require limiting L for the starting routes in Algorithm 4, something which is explored in section 4.4.

Algorithm 6: Update Route

Input: initial_route, all_nodes, scores, info, unexpected, L, zones, zone_id

```
graph = subgraph(all_nodes, zones[[zone_id])
distances = dist(graph)
L_remaining = L - route_length(initial_route)
remaining_route = initial_route
remaining_nodes = c(remaining_route[3:length(remaining_route)])
route = remaining_route[1:2]
while (length(remaining_route) != 0) do
    id_now = tail(route, 1)
    score[id_now] = 0
    candidates = all_nodes !in route
    if (unexpected[id_now]) then
        related_nodes = which(info[id_now,] != 0)
        for (id in related_nodes) do
            if (id !in route) score[id] = score[id] + info[id_now,id]
        unexpected[id_now] = FALSE
    # Track the number of nodes outside the zone that would have been the best candidate
    if (find_best_candidate(graph, distances, id_now) !in zones[[zone_id]]) then
        candidate_outside = candidate_outside + 1
    best_candidate = find_best_candidate(graph, distances, id_now)
    # If no better candidate exists follow the planned route
    if (best_candidate in remaining_route) then
        best_candidate = remaining_route[1]

    # Update the route, planned route and range budget
    route = append(route, best_candidate)
    remaining_route = c(shortest_path(best_candidate, remaining_route[2], all_nodes,
     remaining_route[-(1:2)])
    L_remaining = L - route_length(route)
    if (remaining_route[1] == 1) then
        route = append(route, 1)
        # Exit the while loop when arriving back at the base
        remaining_route = integer(0)
return route, L, L_remaining
```

Algorithm 7: Find Best Candidate

Input: graph, distances, id_now

```
# Find nodes directly connected to id_now
all_nghbrs = neighborhood(graph, order = 1, nodes = id_now)
# Remove nodes either already visited or planned to visit later
nghbrs = all_nghbrs[(all_nghbrs != id_now) & ((all_nghbrs !in route) | (all_nghbrs in
  remaining_route))]

# Find which neighbors can feasibly replace next planned node
L_removed = distances[id_now, remaining_route[1]] +
  distances[remaining_route[1],remaining_route[2]]
L_added = distances[id_now, nghbrs] + distances[nghbrs, remaining_route[2]]
delta = L_added - L_removed
L_added = L_added[delta < L_remaining]
feasible_nghbrs = nghbrs[nghbrs in L_added]

# Find score for the shortest path to each candidate
temp_score = sapply(feasible_nghbrs, function(nghbr){
    temp_path = shortest_path(nghbr, remaining_route[2], graph)
    ids = c(nghbr, temp_path[-length(temp_path)])
    sum(score[ids])
})
sdr = temp_score / L_added
best_candidate = which.max(sdr)
return best_candidate
```

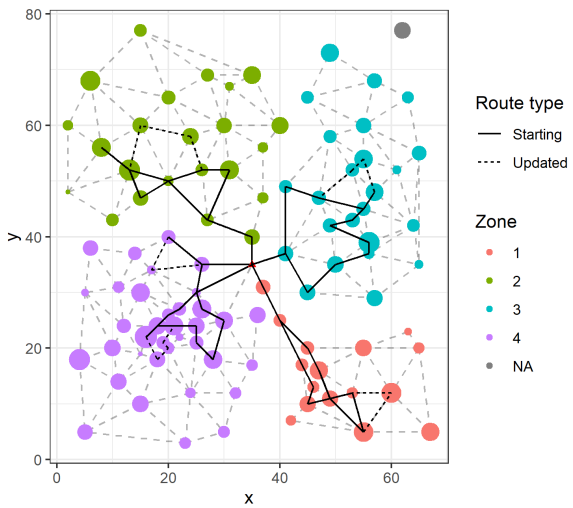


Figure 3.9: Example of updated routes generated by Algorithm 6 and the starting routes they were based on.

4 Experiments

In this chapter experiments using the solution methods presented in Chapter 3 will be conducted, so that they can be compared against both each other and the more common TOP solution method. In order to meaningfully compare the zoning solutions, measures must be introduced for this purpose. The most intuitive measure of performance is the cumulative score, as this in the objective function in the ZTOP and all related problems relevant to this project.

Measures more specifically relevant for evaluating zoning performance will also be considered. These aim to capture how other aspects of the zones, such as routing flexibility during the mission, will influence their applicability. But first the instance used to conduct these experiments will be explained further.

4.1 Experiment Design

This section will describe the characteristics of the experiments, that will be performed in order to evaluate the performance of the zoning methods.

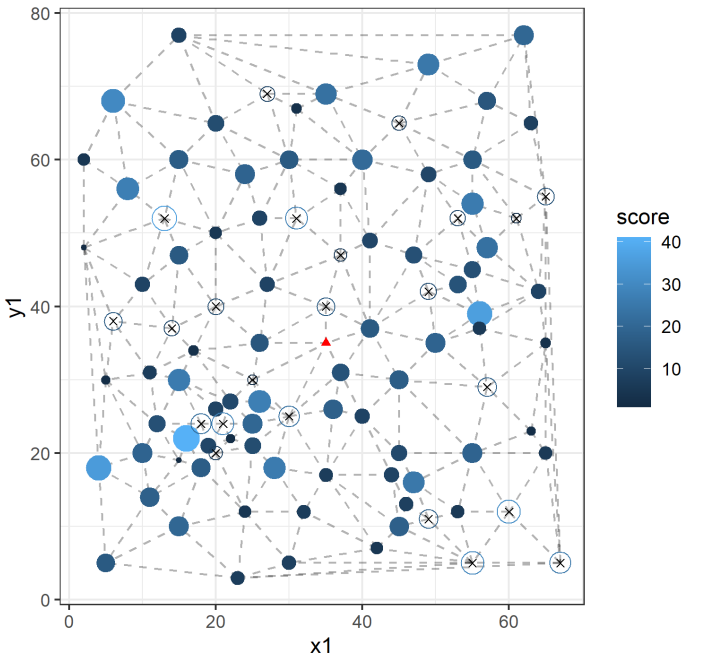
Because all of the zoning methods simply return k subsets of the nodes from the original graph, the same routing methods and implementations can be used for all of them. The next consideration is the graphs itself. Here we have used common examples from literature, as introduced in section 2.1.3. The instance parameters are given in Table 4.1a, where the names are consistent with Chao et al. [8]. The lower range parameter combinations from the original paper have been ignored as zoning is deemed unnecessary in this case, due to the low likelihood of intersecting path and simple ways to solve them. For each of these instances, 50 sets of observations of unexpected events have been drawn from the Bernoulli distribution, as described in section 2.1 under the dynamic aspects, resulting in 50 potentially different route updates. An example of realized unexpected events can be seen in Figure 4.1b. These observations will then be used in the following to evaluate different aspects of each zoning methods solutions.

4.2 Comparison of Zoning Solution Methods

In this section the two zoning methods will be compared in terms of various metrics, in order to determine their effectiveness. This should also give insight into each methods advantages and disadvantages, potentially with a clue as to how the shortcomings can be rectified. In the following the three solution methods introduced in Chapter 3 will be refered to as routing-based (RB) for the approach described in subsection 3.1.2, heuristic-compact (HC) for the one with the objective function seen in Equation (3.1) and likewise for heuristic-relevancy (HR) and Equation (3.2).

	# agents		
	2	*3*	*4*
7.j	100	66.7	50
7.k	110	73.3	55
7.l	120	80.0	60
7.m	130	86.7	65
7.n	140	93.3	70
7.o	150	100.0	75
7.p	160	106.7	80
7.q	170	113.3	85
7.r	180	120.0	90
7.s	190	126.7	95
7.t	200	133.3	100

(a) Combinations of range constraints and number of agents. Each row corresponds to a given range constraint while each column specifies how much range per agent the giving range constraint allows.



(b) The distribution of nodes with both size and color indicating the score, except for the red node which is the source. An example of realized unexpected events is given and the nodes where unexpected event occur is marked with an X.

Figure 4.1: The set of instances used for performance evaluation.

4.2.1 Cumulative route score

As a general comparison of the solution methods presented in Chapter 3, the cumulative score of the starting routes as a function of the length constraint and zoning method can be seen in Figure 4.2. A similar plot can be seen for the updated routes in Figure 4.3, where these are the mean scores over the 50 instances with observations of unexpected events. The associated intervals of cumulative score values are illustrated in Figure 4.4.

It is clear from these plots that RB clustering performs better in general when compared to the heuristic methods and particularly as the number of agents and range constraints increases. This is probably because the lower values for L and k combine to make the zone sizes less relevant, since the restrictions the zones impose are then less impactful.

To better understand the relationship between L and cumulative score, some exemplary solutions have been plotted in Figure 4.5. For 2 agents the RB method seen in A allows the bottom left agent to visit more nodes on its way to the dense area, whereas the HC method, seen in D, requires longer transit time before visiting the relevant nodes. In this case the routing does not take much advantage of it, but it does give potential for higher cumulative score for especially lower range constraints problems. So even though not all nodes in the zone have been visited, the zone could be interpreted as either being too small or having too many of its nodes too far away from the base for optimal performance. This performance gap can therefore potentially be reduced or eliminated by introducing a parameter to limit the cardinality difference between zones in this heuristic approach.

Returning to Figure 4.3 and comparing RB with HC, RB clustering seems to consistently perform better for nearly all parameter combinations. The effect of choosing between using these zoning method is also statistically significant, as can be seen in Table 4.1. This table contains a summary of the routes cumulative scores regressed against a dummy variable indicating zoning method. This shows that using the routing-based clustering method instead of the compact heuristic based meth-

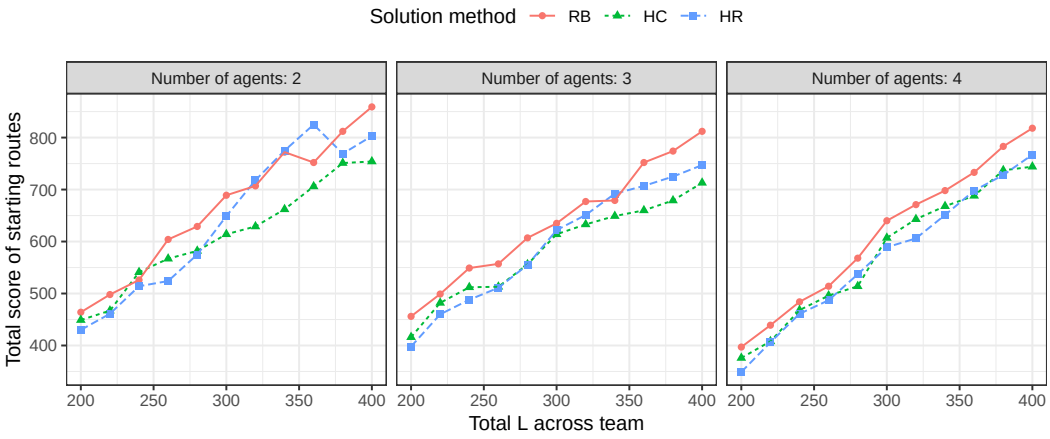


Figure 4.2: The cumulative score of each starting route as a function of the length constraint L, zoning method (color) and the number of agents.

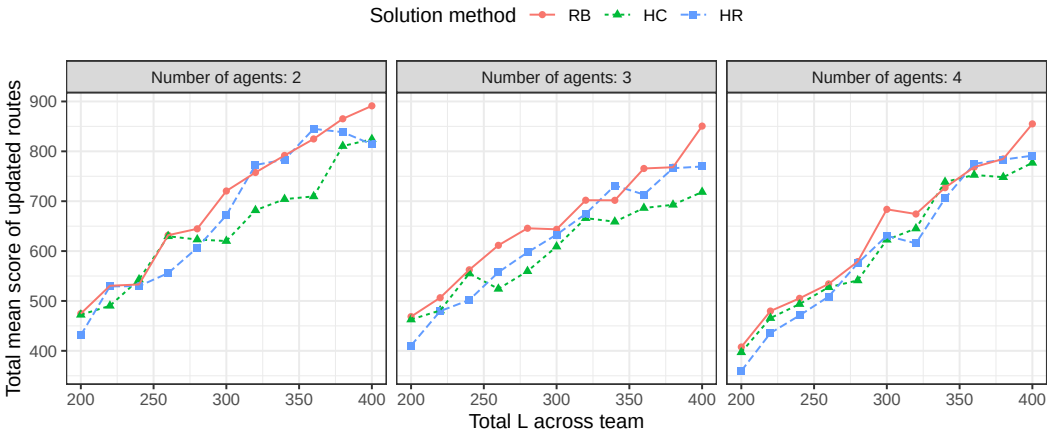


Figure 4.3: The mean cumulative score of the 50 update route as a function of the length constraint L, zoning method (color) and the number of agents.

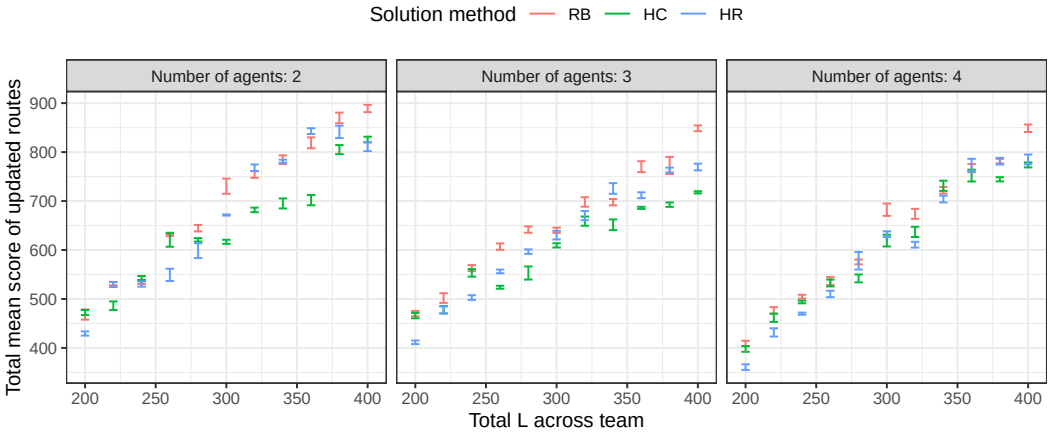


Figure 4.4: The interval of cumulative scores of the 50 update route as a function of the length constraint L, zoning method (color) and the number of agents.

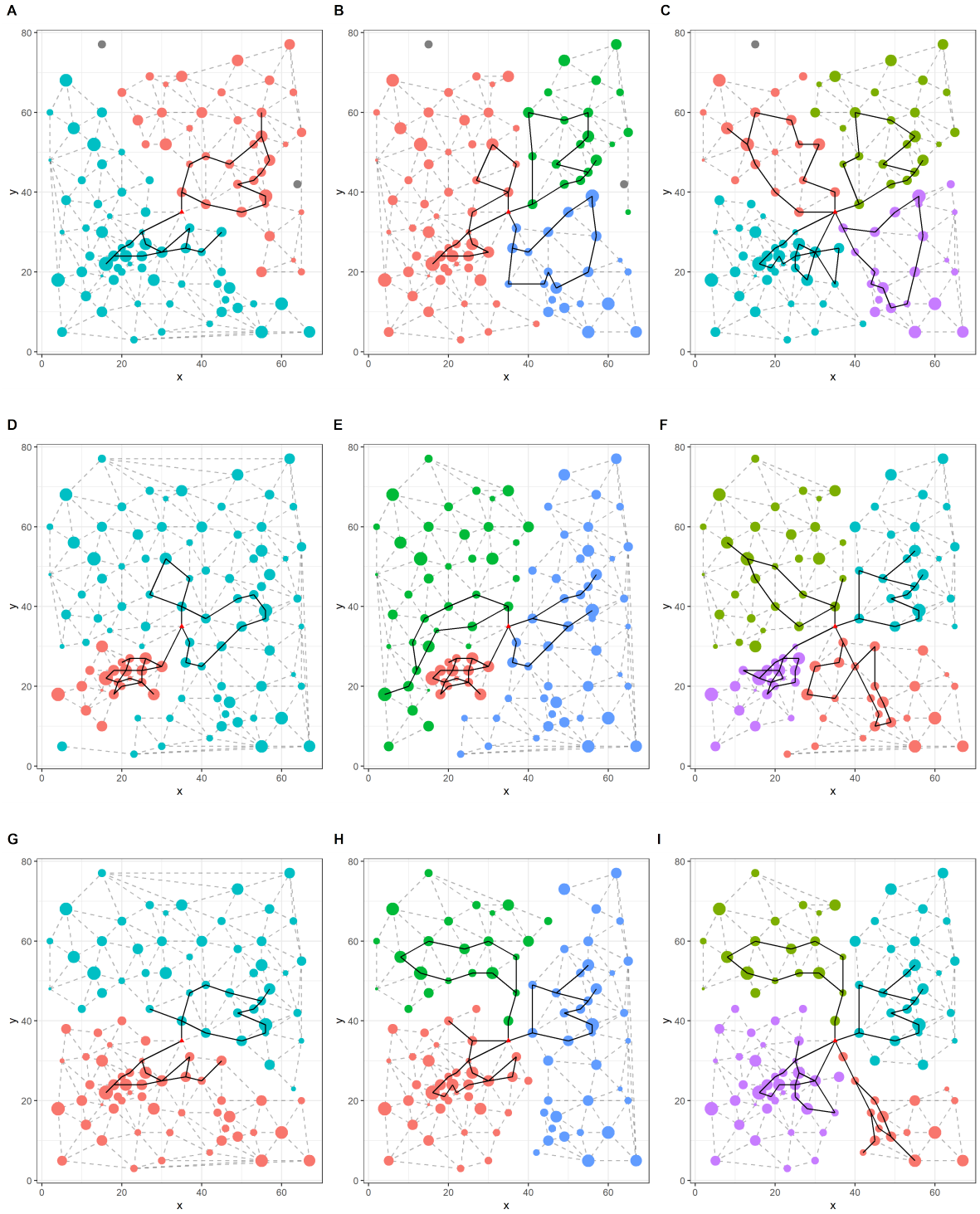


Figure 4.5: Cluster solution examples with updated routes, where each agent has a range of 100. Each row correspond to a different solution method; ABC are obtained from routing-based clustering, DEF are obtained from the compactness heuristic while GHI are obtained from the relevancy heuristic. The columns corresponds to number of agents.

ods significantly increases score of updated routes by around 40 points. The significant improvement over HR is not directly evident in the table but can also be verified.

	Estimate	Std. Error	t value	Pr(> |t|)
(Intercept)	656.2055	3.5230	186.26	0.0000
clustering_method heuristic relevancy	9.0969	4.9823	1.83	0.0679
clustering_method routing-based	41.9004	4.9823	8.41	0.0000

Table 4.1: Linear regression for clustering methods effect on the update routes cumulative score. Uses data for all number of agents, $k = 2, 3, 4$, and length constraints $L = 200, 220, \ldots, 380, 400$.

Regarding the comparison between the heuristic based zoning methods, we see only minor differences in aggregate, but HR seems to suffer less as more nodes can be reached. This was also expected from the discussion above regarding zone size differences and is further supported by the examples in Figure 4.5.

4.2.2 Routing flexibility within the cluster

This subsection seeks to investigate how effectively Algorithm 6 (update routing), can be used for each of the zoning methods. We therefore wish to measure not just the number of alternative nodes available to each route, but how often they are likely to be the best candidate as a result of the changes in scores during the mission. For this purpose we will use the measure of "number of times where the highest SDR node is outside the zone, for which routing is currently being done", described in Algorithm 6 lines 16-17. Given that this reflects how often the desired candidate node is unavailable, this should be a good proxy for how much useful flexibility the clusters allow the update route algorithm to have.

This was explicitly done for the HR approach by considering the information matrix that describes the nodes score dependencies. The RB clustering method on the other hand only does this indirectly by having similar routes in the same zone as alternatives that are also within the cluster. Good alternatives should therefore be available, but they do not account for the nodes score dependencies directly. The last method, HC, does none of these and we therefore expect negligible score improvement from starting to updated routes.

Similarly to the previous subsection, 50 realization of unexpected events are used to regress the number of best candidates outside the zone against the solution method indicator variable. The results can be seen in Table 4.2.

	Estimate	Std. Error	t value	Pr(> |t|)
(Intercept)	3.2218	0.0553	58.25	0.0000
clustering_methodheuristic relevancy	-1.0624	0.0782	-13.58	0.0000
clustering_methodrouting-based	-1.0588	0.0782	-13.54	0.0000

Table 4.2: Linear regression for zoning methods effect on the number of best candidates outside the zone. Uses data for all number of agents, $k = 2, 3, 4$, and length constraints $L = 200, 220, \ldots, 380, 400$.

Here we see fewer best candidates outside the zone for both the RB and HR clustering methods when compared to HC. The cause is again likely to be the simple lack of alternative nodes in the HC solutions. When restricting to $L < 300$ the routing-based clustering methods still leads to fewer best candidates outside the zone, but HR actually has more than HC. The regression results for $L < 300$ and all number of agents is given in Table 4.3. This can be explained by HR not accounting for the routes that would actually be used. It is therefore only good when space is limited and alternative nodes being more relevant, such as for longer routes or ones that are closer to zone boundaries. A more nuanced view of this behavior can be seen in Figure 4.6.

Another measure for route updating flexibility alluded to earlier, is the performance gap between the

	Estimate	Std. Error	t value	Pr(> |t|)
(Intercept)	1.4813	0.0512	28.95	0.0000
clustering_methodHC	-0.3973	0.0724	-5.49	0.0000
clustering_methodHR	0.3573	0.0724	4.94	0.0000

Table 4.3: Linear regression for zoning methods effect on the number of best candidates outside the zone. Uses data for all number of agents, $k = 2, 3, 4$, and length constraints $L = 200, \ldots, 280$.

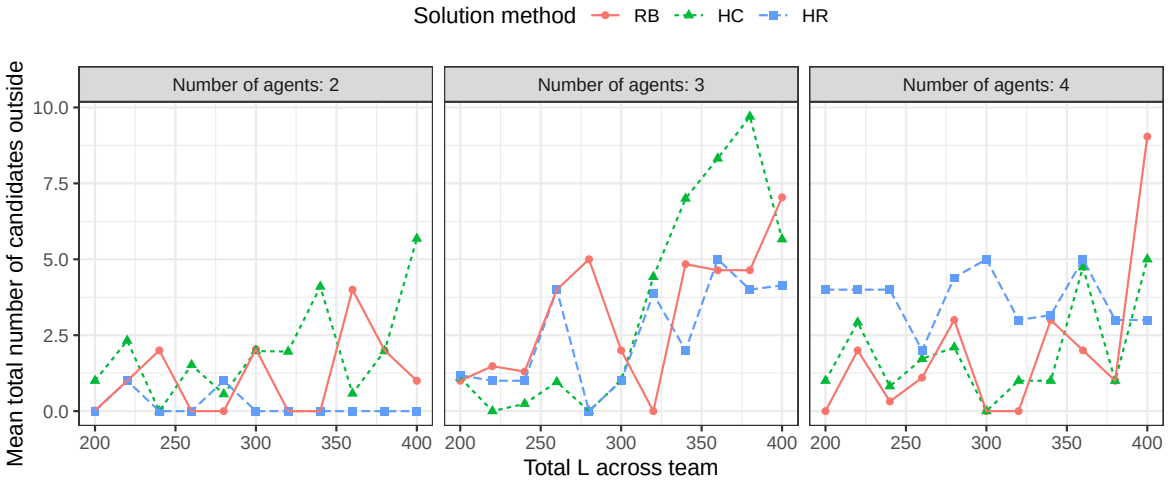


Figure 4.6: The number of best candidates outside the zone of the update route as a function of the length constraint L, zoning method (color) and the number of agents.

starting or improved routes and the updated routes. This should be a good gauge of how effective the zoning allows the updating algorithm to be. This is illustrated for the three zoning methods in Figure 4.7.

Here we see behavior mostly consistent with the results seen above, where HC does nothing to ensure updating flexibility and therefore has inconsistent results. The slight dip around 3 agents is a result of no change when updating the routes, but slightly lower update scores than before the mission The route is not actually updated to a worse alternative, there simply is no better alternative found, in spite of the low realized scores. HR is underwhelming and with small improvements considering it being optimized for having access to relevant nodes for updating. The ability for RB to have both good routes initially and after updates, when compared to the other methods, is however exemplified here consistent with results previously presented.

The general single digit percentage improvements seen from updating indicates that either the dynamic scores have a generally small impact or not enough range is conserved for updating. The latter is supported by the poor performance of HR, but relatively small score updates or limited extra range could explain the general trend. Both of these aspects are however case dependent and the appropriate zoning methods may therefore change on a case by case basis. This is further investigated in section 4.4.

4.3 Comparison with the Team Orienteering Problem

The purpose of this section is to compare the solutions from the ZTOP with conventional TOP solutions, in order to gauge the potential performance degradation caused by the constraint of not having intersecting edges. Based on the performance seen in Khemakhem et al. [18], we expect that this will not be severe and could even be useful in leading the agents to their own areas that they

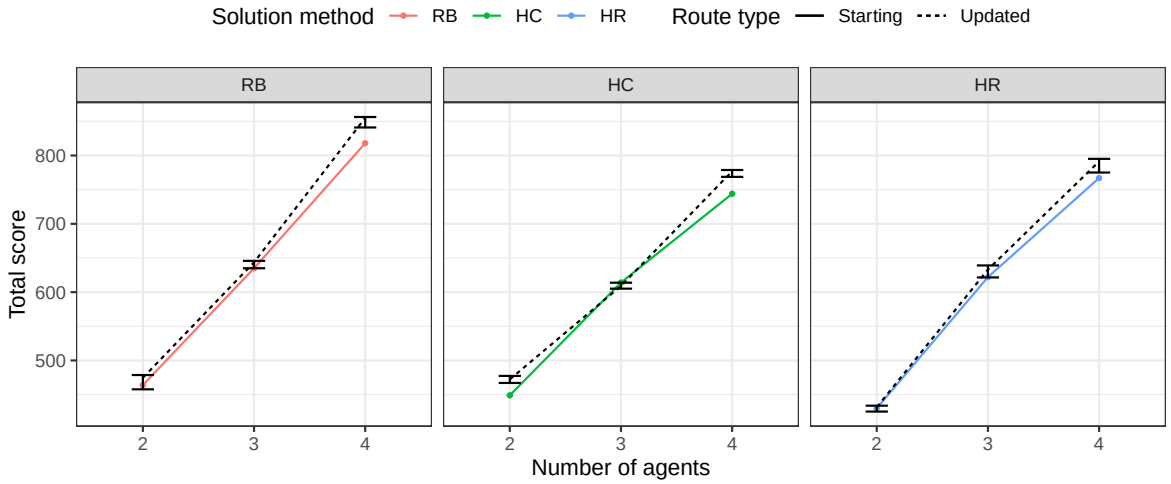


Figure 4.7: The mean cumulative score of the 50 updated routes (dashed) and the cumulative score of improved routes (full), as a function of the number of agents.

can on their own without having some nodes taken by other agents.

In order to compare our zoned approach with the TOP commonly seen in literature, a scheme for using the TOP will be introduced here. One route will be constructed at a time and these nodes will be removed from the graph when considering the next agents route. Additionally, new edges will be added after each route, by doing Delaunay triangulation as before, in order to ensure sufficient connectivity. Importantly this no longer ensure collision free routes, but without these additional edges solution quality is highly degraded or solutions even become infeasible.

Additionally, because no communication between agents during the mission is allowed to influence the score, the routes from the TOP will not be updated, since we cannot ensure that the added node has not been visited by another agent, thereby making the score effectively 0 for one of the agents. The route updating will therefore be used purely to update the scores in order to get during mission performance that can be compared with the ZTOP solutions. Since no route updating is done, all the range will be used in improve routing for the TOP solution method.

Updated routing performance is compared for the four methods in Figure 4.9. On first viewing the performance is clearly in line with the other methods, which confirms our expectations at least when using the routing algorithms described in the report. On closer inspection we see that RB pretty consistently outperforms it, with no systematic disadvantages. This is further verified when looking at regression results similar to the previous section, but between the TOP approach and each of the zoning approaches. This is summarised in Table 4.4. These results are also generally true for regressions with a specific number of agents.

	Estimate	Std. Error	t value	Pr(> |t|)
(Intercept)	625.4859	3.0638	204.15	0.0000
clustering_methodRB	37.9626	4.3329	8.76	0.0000
clustering_methodHC	-6.0312	4.3329	-1.39	0.1640
clustering_methodHR	6.7945	4.3329	1.57	0.1169

Table 4.4: Regression for cumulative updated routing score as a function of solution method including the non-zoned TOP approach.

To further compare RB clustering and the TOP solution method, solution examples with the same parameters are illustrated in Figure 4.8. Clearly the TOP method has overlapping routes, leading to routes passing, or nearly so, over nodes visited by other agents resulting in inefficient routes. A

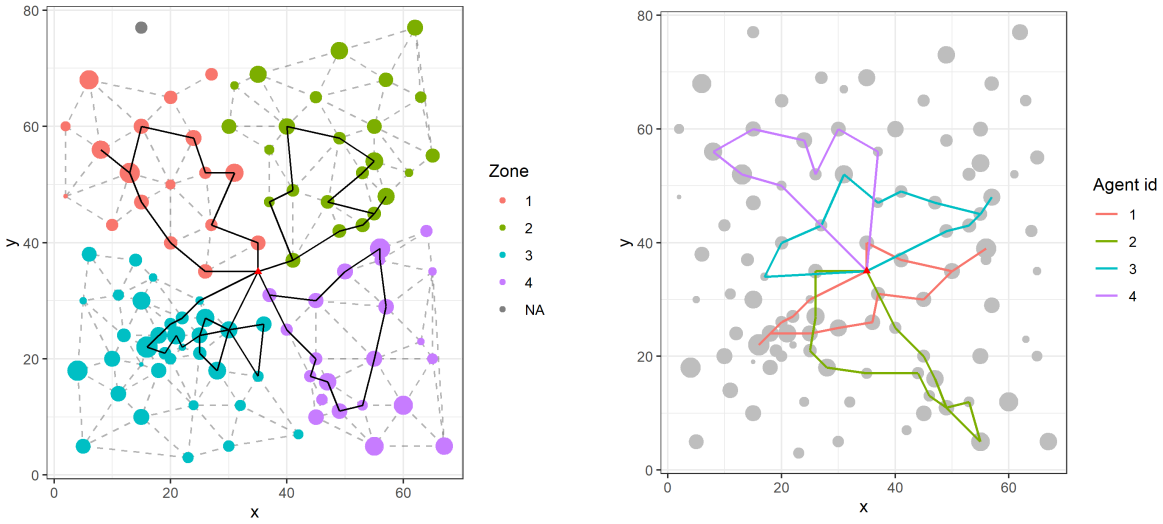


Figure 4.8: A comparison of the TOP and ZTOP approach, using the RB clustering method, for a team of 4 agents with $L = 100$.

less obvious aspect is the tendency to stay around the base instead of exploring further. This does not have a big impact on this map specifically, but may be problematic for larger maps or range constraints.

The obvious cause for staying closer to the base is that the SDR measure considers the length of route home when evaluating each node, thereby discouraging exploration far from the current node. This is mitigated in zoning where the feasible nodes are so limited that more of the map gets explored. This problem may therefore also be mitigated by simply using another routing method that emphasises exploration.

4.4 Reserving Range for Updating Routes

This section will be a brief exploration of the impact of purposely saving range beyond the construction of starting routes can have on performance. A range constraint for each improvement step will therefore be given and performance will be compared to the results from previous sections.

The setup will be such that 60% of the total range L will be available when constructing the starting routes in Algorithm 4. An additional $0.2 \cdot L$ will be available in Algorithm 5 for improving these routes. The full range will finally be available for route updating in Algorithm 6.

An illustration of the difference between starting and updated routes can be seen in Figure 4.10, similar to Figure 4.7 for the original approach. Clearly there is a significant difference, proving that route improvement and updating can use the extra range if required, but it does not show how effectively it is used. Worth noting is that the relative performance between the zoning methods has not changed when going from the previous range allocation approach to this one.

For a more telling comparison we look at Figure 4.11, where the cumulative updated route score of the original, from the previous sections, is compared with this range reserved approach. For this figure the performance across number of agents have been averaged, giving a more intuitive, but slightly skewed interpretation. There is a general trend for better performance as the total range constraint increases, when reserving range for improve and updating routes, but results do vary across method and the number of agents. Different combinations of range reserves for each step in the route construction therefore seems to be worth looking further into.

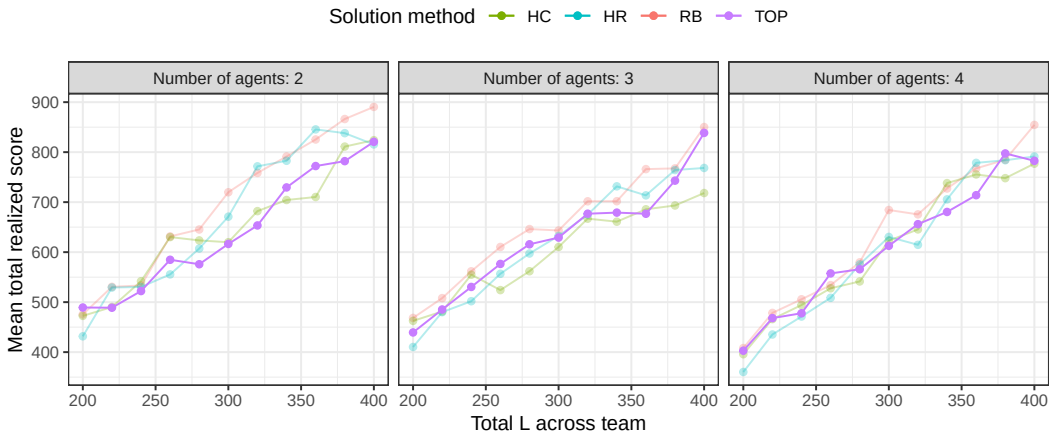


Figure 4.9: Mean total realized scores from 50 realizations of unexpected events across the TOP and ZTOP solution methods.

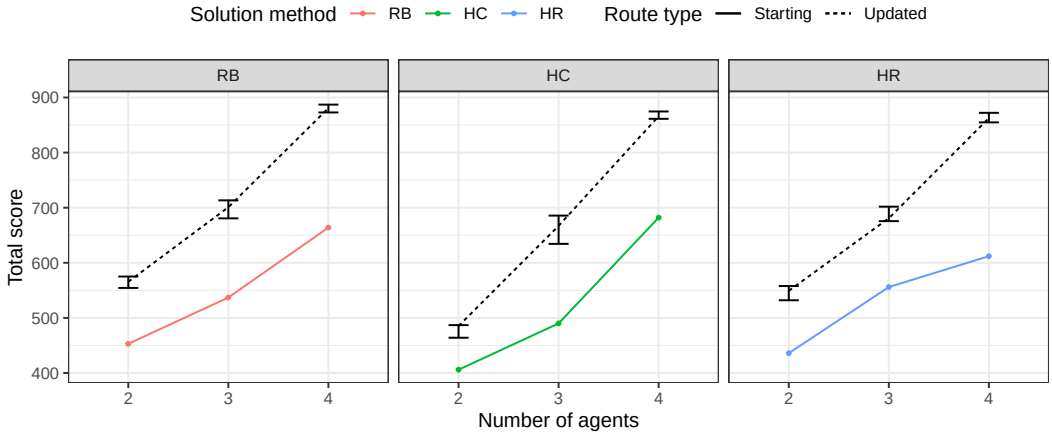


Figure 4.10: The mean cumulative score of the 50 updated routes (dashed) and the cumulative score of improved routes (full), as a function of the number of agents for the range reserved approach.

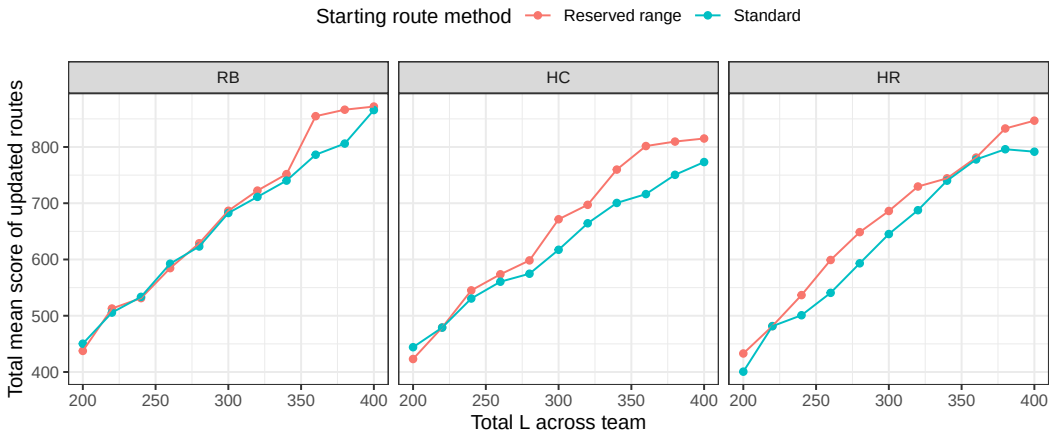


Figure 4.11: Mean total realized scores from 50 realizations of unexpected events across the ZTOP solution methods. Comparison between the solution method used in most of the report (Standard) and the range reserved approach introduced in section 4.4 (Reserved range).

The caveat to this, is that only a simple starting route algorithm is considered and as seen for the RB zoning approach, when starting routes can be made more effectively, using much of the range on them seems to not be an issue. One could imagine that if the routes constructed pre-mission were robust to score changes and of higher quality, updates would only require little range to adjust, unless the environment was exceedingly volatile.

4.5 Conclusion

This chapter has explored different performance aspects of the zoning solutions based on methods presented in Chapter 3.

The ZTOP solution methods were compared in section 4.2, where the cumulative score and so called updating flexibility were evaluated. In terms of cumulative score for both starting and update routes, the best zoning method was found to be the routing-based approach, RB, in all scenarios considered. For updating flexibility, defined primarily as the improvement in cumulative score, from starting or improved routes to updated routes, we saw minor improvements across all zoning methods, but most consistent and highest overall for RB. The number of times the best candidate, evaluated in terms of SDR, was outside the zone for which routing was currently being done, was also mostly consistent with these results. This meant poor performance for HC and more variable but good average performance for HR, as was to be expected from their respective objective functions.

Additionally, a simple comparison with a more traditional TOP formulation was conducted, where favorable results were found, for especially the routing-based approach. More specifically, the regression results from Table 4.4, showed that RB zoning outperformed the TOP approach that was not constraint to zones.

Lastly, the influence of purposely limiting the range used by the starting and improved routes, allowing for more flexible route updating was tested. Just as good or better performance was found for all three zoning methods, indicating that this is a parameter worth considering.

5 Discussion

This chapter will serve to further explore the assumptions and decision made regarding especially the solution methods and testing methodology. Some alternative will therefore be considered in short form, in order to determine the potential and direction for future work on related topics.

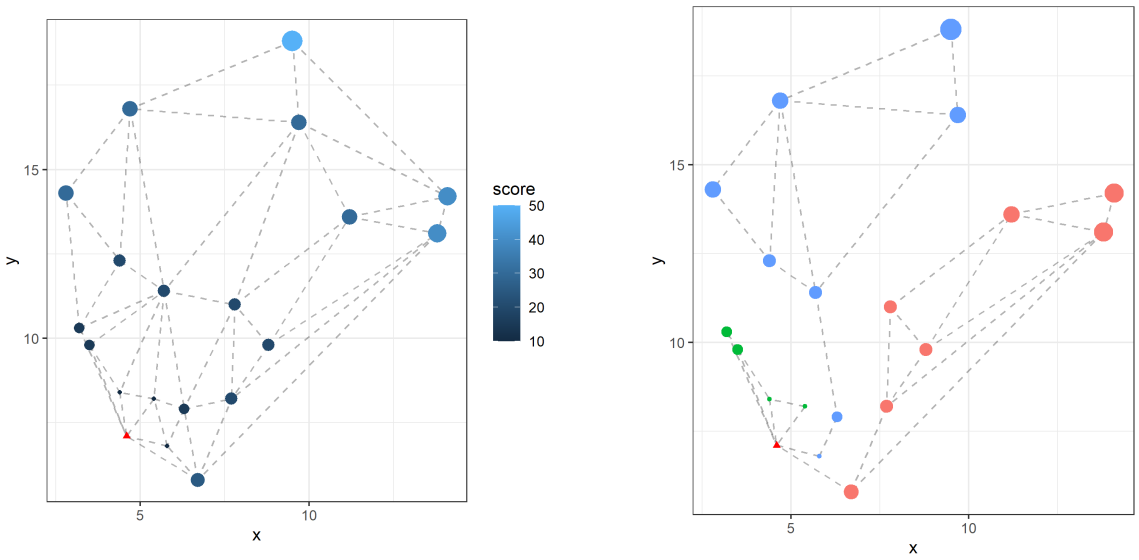
The first heuristic based method, HC, inspired by Khemakhem et al. [18], left out some aspects from the original paper that, based on the results from Chapter 4, seem to have significantly degraded performance. The main contributor is the lack of a limit on cardinality difference between clusters and the subsequent effect this has on routing flexibility. This was particularly evident on the bottom left corner seen in Figure 4.5 of the test instance, where time required before visiting a node was high and the number of nodes to visit low. These combined to reduce score for both the low and high range constraints respectively. As shown by especially the other heuristic methods, HR, simply having more nodes and therefore alternatives both for the starting route and when updating aids performance, despite not explicitly considering starting route performance.

Given the two heuristic clustering approaches different strengths, it would be obvious to try and combine them, to get both the route focused approach of one and the updating flexibility of the other. Since the clustering improvement procedure is the same for both of them, it can be done as a multi-objective function, that could be evaluated as the others were individually. Multi-objective functions is a large area of research in itself, but there are two simple approaches. A linear combination - additive or area/volume where they are multiplied together. The later has the advantage of weighting the importance of each objective equally without requiring any scaling and is therefore easy to implement if this is the desired effect. It does however lack the flexibility of a linear combination, wherefore many alternative exist. Given that they both seem outclassed by the routing based approach, this only seems relevant if differing routing methods significantly change the results found in Chapter 4.

The routing based clustering method also has avenues of potential improvement. A simple change is to consider an alternate similarity measure in hierarchical clustering, that better reflects if updating one route would cause it to become identical or very similar to another. This would then cluster routes such that updating could be more effective. One idea is to use the existing measure from Equation (3.5) and modify it such that more importance is placed on the first part of the route. Greater similarity would then mean more likely to have the same or very similar starting routes and only differing for the part of the route that can be updated. This could therefore improve both starting and updating route performance, without changing the routes themselves.

Regarding the routing solutions, they have been very simple, in order to reduce time spent on computation and implementation, in order to instead focus on the clustering methods. Given the routes proclivity to loop in on themselves, especially where nodes are closer together, there are obvious areas of improvement. Post-processing procedure for the starting or improved routes are most obvious, given their extension to consider removing/swapping nodes in the route or fixing the loops with e.g. 2-Opt or other well known solution methods associated with the TSP or OP.

In the pre-mission step, where scores are constant throughout the routing process, the order in which nodes are visited does not affect the cumulative score, which is the main measure for comparing



(a) Nodes from instance p2 from Chao, with edges using Delaney triangulation.

(b) Clustering solution found by using the compact heuristic solution method.

Figure 5.1: Examples of zones created based on an instance with of center base.

solutions. There may however be value in visiting the highest uncertainty nodes first, in order to have the most flexibility, in terms of remaining range, when these are visited. This dynamic is not considered for the starting or improved routes, but may help improve route updating during the mission.

We cannot be sure what effect changing the routing methods will have on the relative performance of the clustering algorithms, but given the consistency we have seen for the limited scope of implementations we have tried, it seems that the performance rankings hold true regardless of routing method. When combining this with the above changes to the heuristic clustering methods however, the consequences become too uncertain to meaningfully comment on.

The main instance example used throughout this report is particularly conducive to zoning. The main aspect that contributes to this is the centrally located base, that gives the most possible space for the routes not to overlap. This central location therefore makes the additional constraints in ZTOP, as compared to TOP, less restrictive. To illustrate this limitation, clustering solutions on instances with a corner placed base are plotted in Figure 5.1.

This makes the zones look more like corridors and is obviously inefficient when much of the route out from the base is identical to the one that must necessarily be used to get back. Some of this effect can be removed by strategically introducing additional edges, but still results in routes with long transit time relative to the time spent exploring score dense areas. It is important to note here that better zoning solutions for the illustrated instance do exist and can somewhat mitigate the problem, i.e., zones where sufficiently large cycles can be made.

A more favorable scenario would be having nodes at the opposite ends of the graph, such that the corridor shapes are not necessarily a disadvantage. It may also be possible to have a less strict zoning policy, where the UAVs fly to a zone not directly connected to the base, during which their safety is ensured some other way, such as delayed deployment, differing altitudes, etc. This eliminates the need for distinct non-overlapping connections to the base or bases, allowing for denser zones at the cost of added transit time. These are application dependent and therefore not something that will be discussed further here, but is important to note when considering solution methods for a given application.

The edge construction has already been briefly mentioned, but deserves more attention, as this

fundamentally changes the problem and therefore the performance of all the solution methods used. Delaunay triangulation was chosen for its ability to ensure that resulting graphs have no overlapping edges and a high level of connectivity. After the clustering is performed however, the first property becomes less relevant and we are then only concerned with the edges outside the convex hull of the zones, as this is where interactions between agents from different zones could occur. No methods to add edges satisfying this criteria have been explored in this project, but do exist and could influence routing performance significantly and thereby have some of the effects previously described for routing improvement.

Another area of potential improvement is the routing scheme introduced to solve the TOP, with available resource comparable to the ZTOP solution methods. Other possibly better performing schemes have not been explored and may more effectively distribute the nodes between agents such that less range would be consumed to visit the same nodes, thereby allowing for further exploration. One such scheme is to have each agent chose only one node before letting the next agent chose one. This can be seen in e.g., Chao [9], where it often produces the desired routing behavior. However, using this approach instead would likely lead to routing solutions that are even more similar to the zoning approaches, as it can be interpreted as a kind of light zoning or rather grouping, that make up the routes.

In Chapter 4 only 2, 3 and 4 agents were considered, which like the related range constraints and instances was inspired by Chao [9]. More UAVs might however be even more relevant for the ZTOP, as this introduces more potential points of conflict and potential routing inefficiency due to close nodes being visited by different agents. Because of the instance sizes and focus on simplicity these were not explored, but would be relevant for future work given relevant applications.

6 Conclusion

This project started by considering world issues where UAV path planning could or have been used to great effect. A prominent area of research is the increasing scale of forest fires and how they can be monitored, leading to questions regarding aspects of them that may not have been sufficiently accounted for. Based on this we decided to consider dynamic environments under limited communication and how this could be incorporated into existing routing problem formulations.

This lead to the ZTOP formulation introduced in Chapter 2, where the limited communication and safety from collision with other UAVs was accounted for by assigning each UAV to their own zone. The structure of this problem can be seen as a 2-stage stochastic linear programming problem, where the stages refer to zoning and routing respectively and the stochastic element is the how the environment changes during the mission. More accurately, the problem is actually formulated as a multi-stage stochastic problem, where the desirability of visiting a way-point, formalized using scores and nodes respectively, is updated according to the information gathered by the UAV itself. Solving this problem will therefore require distributing the nodes between agents, while considering the distribution of future environmental changes and how it affects the optimal routes for each of these agents.

Solutions to the problem were proposed in Chapter 3, the first of which intended to reflect relevant literature i.e., Khemakhem et al. [18], in the method that we refer to as HC. An important note is that cardinality difference between zones is not directly accounted for, in our implementation unlike the one in Khemakhem et al. [18], leading to the problems discussed in Chapter 5. An altered version of this approach called HR was then proposed to better account for environmental changes during the mission. Finally a so called routing-based approach shortened to RB, was introduced as a way to allow for both near optimal routes pre-mission and flexibility for updates during the mission. This was done by generating a lot of routes using a routing method similar to the one used pre-mission, but with added randomness, and then grouping them using hierarchical clustering. The intention was then that this would lead to good pre-mission routes, while also having good and similar enough alternatives in the zone to adapt to during the mission, if the environment were to change significantly.

These methods were then compared based mainly on their during-mission performance in Chapter 4. Here we found RB to generally outperform the other zoning methods proposed, with the caveats mentioned earlier and further discussed in Chapter 5 These include: Simple routing solution methods not indicative of the best available and the limited variety of graphs considered in this project.

Next, a TOP equivalent of our ZTOP was considered and solved, in order to gauge potential performance degradation when constraining the agents to a subset of the entire graph. This did not seem to be the case for at least the better zoning solutions, such as RB, and might even have helped with each of them exploring their own area, instead of fighting over the same nodes placed in the vicinity of the source node.

The chapter then concludes with looking at the impact of reserving a pre-specified amount of range for each of the routing stages: Starting, improve and updating. The scheme proposed is 60% of the

total available range for starting routes, up to 80% for improvement of the route and then the full 100% for route updating. For this specific case a positive effect on updated route score is seen, when compared to the approach used in the first part of the chapter. This is therefore deemed a relevant parameter for applications of the ZTOP, worthy of further research.

In conclusion, the ZTOP can be effectively solved using the routing-based zoning algorithm described in Section 3.1.2 and shows potential when compared to the more conventional TOP approach. However, the limited graphs, instances and routing algorithms used to obtain these results can not be used to infer general behavior and solution quality. They instead serve as a starting point for exploring the potential of zoning approaches in vehicle routing problems under environmental uncertainty.

Bibliography

[1] Hashnayne Ahmed. Formulation of two-stage stochastic programming with fixed recourse. *Britain International of Exact Sciences (BIoEx) Journal*, 1(1):18–21, 2019.

[2] G. Best and G.A. Hollinger. Decentralised self-organising maps for the online orienteering problem with neighbourhoods. pages 139–141, 2019. doi: 10.1109/MRS.2019.8901053. URL https://www.scopus.com/inward/record.uri?eid=2-s2.0-85075631109&doi=10.1109%2fMRS.2019.8901053&partnerID=40&md5=f06e4928bf75c4d7ae9bb81c6f43f69d.

[3] G. Best and G.A. Hollinger. Decentralised self-organising maps for multi-robot information gathering. pages 4790–4797, 2020. doi: 10.1109/IROS45743.2020.9341106. URL https://www.scopus.com/inward/record.uri?eid=2-s2.0-85102399730&doi=10.1109%2fIROS45743.2020.9341106&partnerID=40&md5=76d471de9ceca150561e7e1e1141c38a.

[4] Mareike Bockholt and Katharina A. Zweig. Clustering of paths in complex networks. *Studies in Computational Intelligence*, 693:183 – 195, 2017. doi: 10.1007/978-3-319-50901-3_15. URL https://www.scopus.com/inward/record.uri?eid=2-s2.0-85007348428&doi=10.1007%2f978-3-319-50901-3_15&partnerID=40&md5=096d43e501def5d13ad3686eed288cd4.

[5] Matthias M Boer, Victor Resco de Dios, and Ross A Bradstock. Unprecedented burn area of australian mega forest fires. *Nature Climate Change*, 10(3):171–172, 2020.

[6] Scott A Bortoff. Path planning for uavs. In *Proceedings of the 2000 American Control Conference. ACC (IEEE Cat. No. 00CH36334)*, volume 1, pages 364–368. IEEE, 2000.

[7] D.W. Casbeer, R.W. Beard, T.W. McLain, Sai-Ming Li, and R.K. Mehra. Forest fire monitoring with multiple small uavs. In *Proceedings of the 2005, American Control Conference, 2005.*, pages 3530–3535 vol. 5, 2005. doi: 10.1109/ACC.2005.1470520.

[8] I.-M. Chao, B.L. Golden, and E.A. Wasil. The team orienteering problem. *European Journal of Operational Research*, 88(3):464–474, 1996. doi: 10.1016/0377-2217(94)00289-4. URL https://www.scopus.com/inward/record.uri?eid=2-s2.0-0030574659&doi=10.1016%2f0377-2217%2894%2900289-4&partnerID=40&md5=1bb7059f7032bc1d9f71da5cabe30f04.

[9] I-Ming Chao. *Algorithms and solutions to multi-level vehicle routing problems*. PhD thesis, University of Maryland, College Park ProQuest Dissertations Publishing, 1993. 9407615., 1993.

[10] Georges A Croes. A method for solving traveling-salesman problems. *Operations research*, 6(6): 791–812, 1958.

[11] Gabor Csardi and Tamas Nepusz. The igraph software package for complex network research. *InterJournal*, Complex Systems:1695, 2006. URL https://igraph.org.

[12] Edsger W Dijkstra et al. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.

[13] K. Erdoğdu and K. Karabulut. Bi-objective green vehicle routing problem. *International Transactions in Operational Research*, 29(3):1602–1626, 2022. doi: 10.1111/itor.13044. URL https://www.scopus.com/inward/record.uri?eid=2-s2.0-85112478667&doi=10.1111%2fitor.13044&partnerID=40&md5=be6f3b90123d1968c2923a0e1ba9da01. cited By 1.

[14] A. Gunawan, H.C. Lau, and P. Vansteenwegen. Orienteering problem: A survey of recent variants, solution approaches and applications. *European Journal of Operational Research*, 255(2): 315–332, 2016. ISSN 03772217. doi: 10.1016/j.ejor.2016.04.059. URL https://www.scopus.com/inward/record.uri?eid=2-s2.0-84969597553&doi=10.1016%2fj.ejor.2016.04.059&partnerID=40&md5=11a464c987e93362330bba9aa23f0fda. cited By 292.

[15] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An introduction to statistical learning*, volume 112. Springer, 2013.

[16] Marcelo Kallmann. Path planning in triangulations. In *Proceedings of the IJCAI workshop on reasoning, representation, and learning in computer games*, pages 49–54, 2005.

[17] Liangjun Ke, Laipeng Zhai, Jing Li, and Felix T.S. Chan. Pareto mimic algorithm: An approach to the team orienteering problem. *Omega*, 61:155–166, 2016. ISSN 0305-0483. doi: https://doi.org/10.1016/j.omega.2015.08.003. URL https://www.sciencedirect.com/science/article/pii/S0305048315001589.

[18] Mahdi Khemakhem, Frédéric Semet, and Habib Chabchoub. A hybrid heuristic for the selective vehicle routing problem. 08 2005. doi: 10.13140/2.1.5044.0644.

[19] G. Laporte and S. Martello. The selective travelling salesman problem. *Discrete Applied Mathematics*, 26(2-3):193–207, 1990. doi: 10.1016/0166-218X(90)90100-Q. URL https://www.scopus.com/inward/record.uri?eid=2-s2.0-0004873816&doi=10.1016%2f0166-218X%2890%2990100-Q&partnerID=40&md5=dc1d086bdb6464393ed8d32a225394d5.

[20] T. Murray, J. Garg, and R. Nagi. Prize collecting multiagent orienteering: Price of anarchy bounds and solution methods. *IEEE Transactions on Automation Science and Engineering*, 19(1): 531–544, 2022. doi: 10.1109/TASE.2020.3041712. URL https://www.scopus.com/inward/record.uri?eid=2-s2.0-85098757629&doi=10.1109%2fTASE.2020.3041712&partnerID=40&md5=6019a2baeab614f0921e5abcac6a5205.

[21] National Interagency Fire Center. Total wildland fires and acres (1983-2021). https://www.nifc.gov/fire-information/statistics/wildfires. Accessed: 2022-06-01.

[22] Javier Panadero, Jesica de Armas, Christine S.M. Currie, and Angel A. Juan. A simheuristic approach for the stochastic team orienteering problem. In *2017 Winter Simulation Conference (WSC)*, pages 3208–3217, 2017. doi: 10.1109/WSC.2017.824803.

[23] C.B. Pedersen, K.G. Nielsen, K. Rosenkrands, A.E. Vasegaard, P. Nielsen, and M. El Yafrani. A grasp-based approach for planning uav-assisted search and rescue missions. *Sensors*, 22(1), 2022. doi: 10.3390/s22010275. URL https://www.scopus.com/inward/record.uri?eid=2-s2.0-85121869011&doi=10.3390%2fs22010275&partnerID=40&md5=cb9a2cf9f0d805db7261a331740695d9.

[24] T. Tsiligirides. Heuristic methods applied to orienteering. *Journal of the Operational Research Society*, 35(9):797–809, 1984. doi: 10.1057/jors.1984.162. URL https://www.scopus.com/inward/record.uri?eid=2-s2.0-84974863294&doi=10.1057%2fjors.1984.162&partnerID=40&md5=7ee65a148a05c9eac7b4b20ae66c1bc7.

[25] P. Vansteenwegen, W. Souffriau, and D.V. Oudheusden. The orienteering problem: A survey. *European Journal of Operational Research*, 209(1):1–10, 2011. ISSN 03772217. doi: 10.1016/j.ejor.2010.03.045. URL https://www.scopus.com/inward/record.uri?eid=2-s2.0-78649467505&doi=10.1016%2fj.ejor.2010.03.045&partnerID=40&md5=59f45323cd9aaea00968e568100ee022. cited By 601.

[26] Hadley Wickham, Mara Averick, Jennifer Bryan, Winston Chang, Lucy D'Agostino McGowan, Romain François, Garrett Grolemund, Alex Hayes, Lionel Henry, Jim Hester, Max Kuhn, Thomas Lin Pedersen, Evan Miller, Stephan Milton Bache, Kirill Müller, Jeroen Ooms, David Robinson, Dana Paige Seidel, Vitalie Spinu, Kohske Takahashi, Davis Vaughan, Claus Wilke, Kara Woo, and Hiroaki Yutani. Welcome to the tidyverse. *Journal of Open Source Software*, 4(43): 1686, 2019. doi: 10.21105/joss.01686.

[27] Y. Zhang, L. Wu, and S. Wang. Ucav path planning by fitness-scaling adaptive chaotic particle swarm optimization. *Mathematical Problems in Engineering*, 2013, 2013. doi: 10.1155/2013/705238. URL `https://www.scopus.com/inward/record.uri?eid=2-s2.0-84881434604&doi=10.1155%2f2013%2f705238&partnerID=40&md5=3946104c7ae3b88459d1afc6c3d03704`. cited By 67.