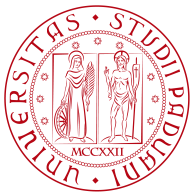


University of Padova

Department of Mathematics

Master Thesis in Data Science

Exact and heuristic algorithms for multi-robots system routing, oriented to underwater monitoring

Supervisor

Professor Francesco Rinaldi

University of Padova

Co-supervisor

Doctor Giulia De Masi

Doctor Emiliano Traversi

Technology Innovation Institute

Master Candidate

Mattia Miolato

Academic Year

2021-2022

Abstract

The exploration of the underwater environment has always been a relevant field for science and technology, to enlarge our knowledge of this mainly unexplored environment.

In this work, we solve an optimization problem for underwater exploration and monitoring based on a fleet of small autonomous underwater vehicles (AUVs). We assume a coarse-grained map is already available from satellite measurements and the set of robots is used to get detailed information on sea bottom features. We provide exact and heuristic integer linear programming methods for finding both the optimal starting position and path planning for a fleet of drones. To obtain a realistic model useful in real applications, we enhance our formulation by imposing connectivity constraints among the AUVs.

Lastly, we present a use case application for coral reef monitoring with real data taken by Abu Dhabi environmental authorities.

Contents

Listing of figures

Listing of tables

Listing of acronyms

AUV . . . . . Autonomous Underwater Vehicle

VRP . . . . . Vehicle Routing Problem

OP . . . . . Orienteering Problem

TOP . . . . . Team Orienteering Problem

MILP . . . . . Mixed Integer Linear Programming

RSESP . . . . . Resource Constraint Elementary Shortest Path

RMP . . . . . Restricted Master Problem

Introduction

Underwater exploration is one of the main technological challenges of the next century. Covering the 70% of the Earth's surface, the oceans represent a big source of mineral and biological resources that can be possible sources of food, energy, and medicines for future generations if managed in a sustainable way. Nevertheless, ocean exploration shows very critical aspects, due to the high pressures, complex hydrodynamics, absence of GPS, and limited communications. For these reasons, exploration missions are usually complex and costly and can be managed by Governmental or transnational authorities.

In this thesis, we propose the use of an underwater fleet for cheaper and more reliable exploration. Compared to the use of a single vehicle, a fleet is cheaper, more flexible, scalable, and robust, and -most relevantly- has improved perception, because it can provide detailed information (images, videos) of large areas even in complex environments.

In our scenario, a detailed (high-resolution) exploration and mapping are performed by a fleet of AUVs. A centralized controller has a map with coarse resolution, as can be obtained from satellite images. Based on this map and on the selection made by the operator about the area to be deeply investigated, a centralized controller runs an optimization algorithm able to identify the best paths each vehicle has to cover, in order to get the most possible information for a specific feature (coral, grass, rock) minimizing the energy consumption and guaranteeing all the robots complete their mission. Then, the controller transfers to each AUV its best path and the map. To get higher-resolution images, a fleet of robots will be used. A central controller has to choose the optimal trajectories for the fleet. To find the best paths for each vehicle, we implement several models and algorithms in the context of Mathematical Optimization. When dealing with a group of robots the identification of the best path to be covered by each robot is in fact an optimization problem belonging to the large class of Routing Problems.

The thesis is organized as follows: in Chapter 1 some basics regarding Mathematical Optimization models are presented. In Chapter 2 the problem is clearly defined and outlined, while the literature is presented. In Chapter 3 an exact model is presented. In Chapter 4 an extended formulation through column generation is presented. In Chapter 5 the extended formulation is enhanced by adding communication constraints between vehicles. The results are summarized in Chapter 6. Lastly, we give an overview of our work and future applications in the

Conclusion.

1 Preliminaries

1.1 Team Orienteering Problem

The Team Orienteering Problem (**TOP**) belongs to the class of routing problems. It is an extension of the Orienteering Problem (**OP**) since we have at our disposal a fleet made by several units that independently can visit customers. This kind of problem takes inspiration from Orienteering, an outdoor sport in which a competitor (or a team) has to start at a specific point and has to visit as many control points as possible in a given time interval. Each control point gives some points if it is visited and the aim of the competitor is to collect as many points as possible and reach the finish point before the time runs out. This competition usually takes place in mountains or forested area, so it is difficult to reach every control point. In fact, the competitor has to choose carefully a subset of point in order to maximize the total score and to be able to reach the end point before it is too late.

TOP problem was presented by Chao et al. [1] and it has been shown to be at least NP-hard. In the following we present a network based representation of TOP and we give also a formulation of the problem. Let's take into consideration a undirected graph $G = (V, E)$, where $V = 1, ..., n$ is the set of vertices and E the set of edges. The first vertex is the starting and the end point of each vehicle's tour and all other nodes are the customers. Each customer is associated with a score s_i ($s_1 = 0$) that is a non-negative quantity. (i, j) is an edge of graph G if it is possible to travel from node i and node j and at each edge is associated a weight c_{ij} that

is the travel time (weights are symmetric in this particular settings). We consider a fleet made by m vehicles, each vehicle can visit any subset of V within a given time limit T_{max}, but a customer can be visited by only one vehicle. Find a solution for TOP means to maximize the total profit collected by the fleet observing the bound imposed by T_{max}. We present a mathematical formulation as in [2]. Let x_{ij}^p be equal to 1 if edge (i, j) belongs to path p, 0 otherwise, y_i^p be equal 1 if vertex i is visited in path p, 0 otherwise and u_i^p be the position of vertex i in path p. We can formulate the model as follow:

$$\max \sum_{p=1}^{m} \sum_{i=1}^{n} s_i y_i^p \tag{1.1}$$

$$\sum_{p=1}^{m} \sum_{j=2}^{n} x_{1j}^p = \sum_{p=1}^{m} \sum_{i=2}^{n-1} x_{i1}^p = m \tag{1.2}$$

$$\sum_{p=1}^{m} y_k^p \leq 1 \qquad \forall k = 2, ..., n-1 \tag{1.3}$$

$$\sum_{i=1}^{n-1} x_{ik}^p = \sum_{j=2}^{n-1} x_{kj}^p = y_k^p \qquad \forall k = 2, ..., n-1; \forall p = 1, ..., m \tag{1.4}$$

$$\sum_{i=1}^{n-1} \sum_{j=2}^{n} c_{ij} x_{ij}^p \leq T_{max} \qquad \forall p = 1, ..., m \tag{1.5}$$

$$2 \leq u_i^p \leq n \qquad \forall i = 1, ..., n; \forall p = 1, ..., m \tag{1.6}$$

$$u_i^p - u_j^p + 1 \leq (n-1)(1 - x_{ij}^p) \qquad \forall i, j = 2, ..., n; \forall p = 1, ..., m \tag{1.7}$$

$$x_{ij}^p, y_i^p \in \{0, 1\} \qquad \forall i, j = 1, ..., n; \forall p = 1, ..., m \tag{1.8}$$

The score function (1.1) maximises the total collected score. Constraints (1.2) require that each path starts and ends in vertex 1. Constraints (1.3) ensure that every vertex is visited at most once. Constraints (1.4) guarantee the connectivity of each path. Constraints (1.5) ensure the max time travel. Constraints (1.6) and (1.7) prevent the presence of sub-tours.

1.2 Branch and Bound

Branch and bound is a solution method that can be applied to many different problem types. A branch and bound algorithm divides the solution space into sub-problems using the divide and

conquer method, then optimizes independently over each sub-problem. When the branch and bound method is applied to an integer programming problem, it is used in conjunction with the normal non-integer solver methods. The method is based on the fact that when a subset is not solved at integrality, only a so-called linear relaxation is solved. In this way, the computation is faster and after evaluating a lower bound (or upper bound, depending on the problem) it is possible to prune a sub-problem if some criteria are met.

Let's say we want to solve a maximization problem. We start by finding through some heuristic a feasible solution that we call $\bar{s}$. This solution will be an initial lower bound for our problem. In fact, if there is a better solution, then its value should be greater than the one we found at the start. Then, we relax the integral constraints, allowing more solution that are not feasible for the initial problem and we solve the problem again. The value of the solution we obtain in this way is an upper bound for our set of feasible solution, since we have enlarged the state space. Now, if the value of lower bound and upper bound are equal, than we already have an optimal solution and the algorithm terminates. Otherwise, we identify n sub-problems such that the union of their state space is equal to the original one. Then, we add this sub-problem to a list of not explored ones. This procedure is called branching. To continue the algorithm, we select one of the candidate sub-problems and process it. There are four possible results. If we find a feasible solution better than $\bar{s}$, then we replace $\bar{s}$ with the new solution and continue. We may also find that the sub-problem has no solutions, in which case we discard (prune) it. Otherwise, we compare the upper bound for the sub-problem to our lower upper bound, given by the value of the best feasible solution encountered thus far. If it is lower than or equal to our current lower bound, then we may again prune the sub-problem. Finally, if we cannot prune the sub-problem, we are forced to branch and add the new sub-problems to the list of active candidates. We continue in this way until the list of candidate sub-problems is empty, at which point our current best solution is, in fact, optimal.

1.3 Column Generation

Column generation is an algorithm for solving mixed integer linear programs with a large (usually exponential) number of variables. The main idea is to use an iterative algorithm that starts with a considerably smaller amount of variables and, at each iteration, either adds a so-called improving variable or proves that the solution found so far is optimal. It is mainly known for its performance on the cutting stock problem [3], but it has been applied to many different problems such as scheduling problems and routing problems.

The algorithm is made by two problems called master problem and subproblem. The master problem keeps track of the actual solution of our model. The aim of the subproblem is to generate a new improving variable to be added to the master problem. The subproblem optimizes over a new domain in which it is possible to build elements represented by a variable in the master problem. This optimization can be performed in different ways, i.e. thanks to MILP, dynamic programming. The master problem with a restricted number of variables is called Restricted Master Problem (**RMP**)

1.4 Branch and Price

The column generation algorithm is a technique used to solve a linear problem. To solve a mathematical formulation containing discrete variables it is necessary to include the column generation algorithm in a branch and bound framework. As the name suggests, the branch and price algorithm is the of combination between branch and bound and column generation.

- The RMP is solved through column generation. Columns are added until it is needed.
- If the solution found is admissible for the MILP, then the algorithm is terminated and the solution is optimal.
- A well designed branch and bound is now applied and at each iteration, it is performed column generation til the branching scheme ends.

Looking deeper into this algorithm, it is noticeable that the decision regarding branching has to be designed carefully. Adding constraints to the RMP can make it impossible to formulate the pricing problem.

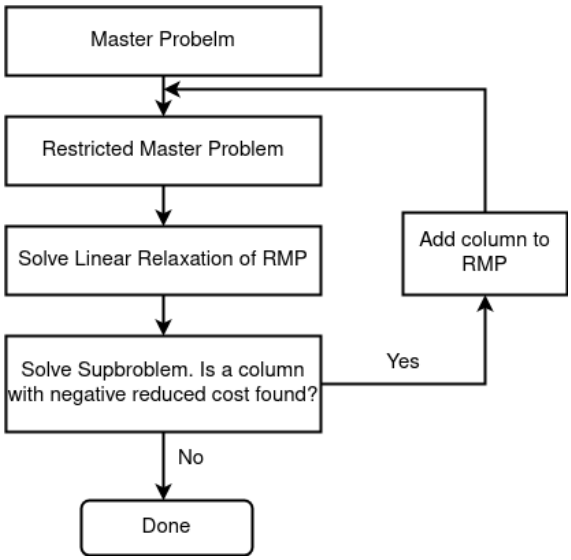


Figure 1.1: High-level outline of the column generation algorithm.

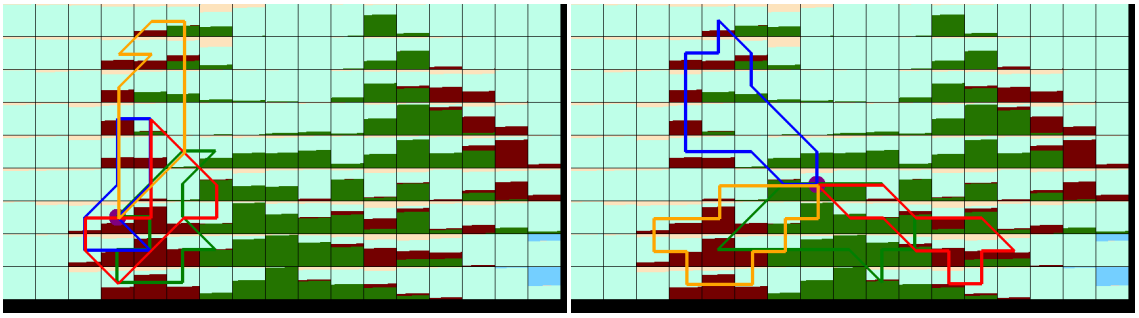
2

Problem Description

2.1 Description of the problem

The basic formulation of our problem is inspired by the team orienteering problem. In our setting, we have a graph where each node has a value between 0 and 1. These values represent the quantity of a feature that a vehicle can collect if it passes through that node. Each edge has a real positive value that represents the distance between the two nodes linked. We have at our disposal a fixed number of vehicles that can explore the graph. Each vehicle has a limited capacity regarding the total distance it can travel. For this reason, in general, it is not possible to explore all the nodes and the model should be able to select only a subset of nodes to be visited. The objective of the problem is to maximize the amount of a feature collected by the fleet and we do not want to minimize the amount of energy consumed by the fleet. What is different from a standard TOP is that we do not know a priori the position of the starting point that we are going to call *base station*. We take as an assumption that the starting and the ending points are the same for all the vehicles. The last two assumptions we make are motivated by the fact that the application underlying this scenario is the exploration of a marine environment. As we are going to present later in this chapter, we are deploying a fleet of robotic fish from a boat, so the starting point of the mission can be chosen carefully to maximize the collection of the desired marine ecosystem feature.

To have a better idea of what is our setting, we show here some examples of the final results.



(a) "Bad" routing.

(b) "Good" routing.

Figure 2.1: Two examples of possible solution.

In figure 2.1, we present two possible solutions for one of our instances. The map contains different colors associated with different features. In this example, we are interested in maximizing the quantity of coral (purple color). We can notice that the position of the base station is different (the purple circle), four vehicles are operating and they are trying to maximize the amount of red feature collected. As we are going to show later, we can see that the choice of the starting point is a critical decision to make, since it influences the mission itself. We refer to the left picture as "bad" routing since the position of the base station is not optimal and it does not allow the vehicles to explore an interesting region. On the other side, we can see that, if the base station is correctly placed, the fleet is able to collect a larger amount of information within the same restriction of autonomy.

2.1.1 Problem variants

Since this work is motivated by real application, we have to consider what could happen in a real scenario and the specification asked by the real use case. We present here two additions that can be useful in a real-life mission.

Communication constraints When we deploy the AUVs in the sea, we want to be sure that we don't lose them. A safety measure we can adopt is to ensure that there is stable communication between every AUV and the base station. Since this requirement is too strict and reduces a lot the explorable region, we can ask for a multi-hop communication between the base station and the vehicles. In Figure 2.2, we can see that the vehicle on the bottom left (each vehicle is a black dot) is not directly communicating with the base station (purple dot), but is connected with another vehicle that is linked with the base station.

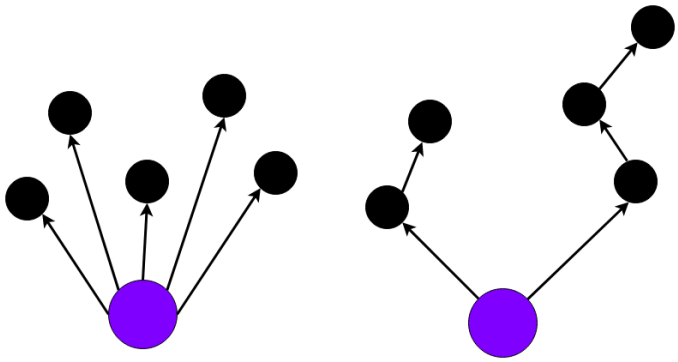


Figure 2.2: Example of multi-hop communication. The base station is purple and the other nodes are black.

Underwater currents When an AUV is following a path, this is influenced by the effect of currents. For this reason, it is not fair to consider a symmetrical graph since moving from one node to another one could be affected by the currents and the energy consumption can be different. To have a more realistic setup, we implement a function that returns the energy consumption based on historical data of currents in the region of interest.

2.2 Data collection

In our real scenario, a first coarse-resolution map is available to a centralized controller. Usually, a map is derived from satellite measurements and it is made available by local environmental authorities. Alternatively, more rarely, a map can result from a long-term exploration and mapping mission performed by an unmanned underwater vehicle. The input data of our work is derived from satellite imagery, where a bathymetry map with soil and vegetation description is provided by the City Environmental Authority. The map distinguishes 3 features out of sand: grass, coral, and rock. This first low-resolution map allows the operator to identify the areas of interest. To have higher-resolution data, a fleet of robots has to be deployed. The exploring AUVs are "small" and equipped with simple sensors. Each one can get close to the sea bottom to get good-resolution images but it can only cover a limited area. Therefore the single robot is unable to solve this task without cooperating with the other AUVs in the fleet. The full fleet, on the contrary, can guarantee the coverage of an extended area, collecting at the same time high-resolution images and videos. For this reason, after an operator selects the area of interest, a centralized controller calculates the deployment point and the optimal path for each robot to maximize the coverage of one or more specific features (rock, coral, sea grass) to be monitored. The controller assigns to each robot the proper path and the low-resolution map.

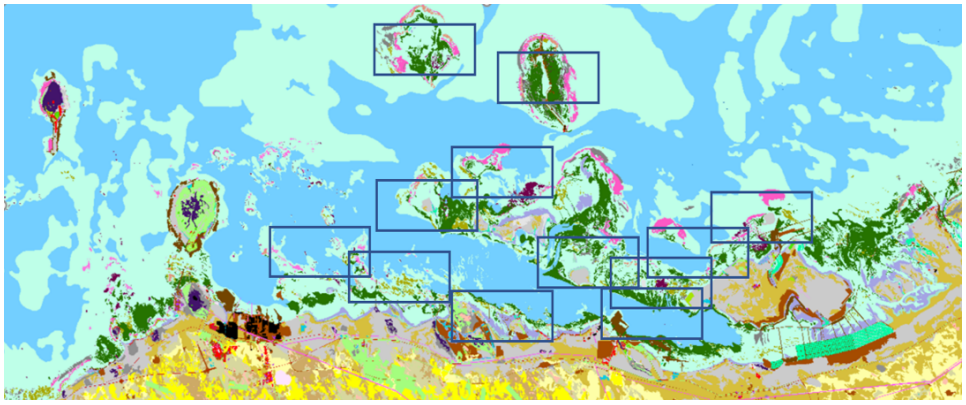


Figure 2.3: Map showing the mainland and marine environment studied and the 12 different snapshots used in the experimental section.

2.2.1 Basic data

The experiments are based on a map provided by the City Environment Agency (see Figure 2.3). The maps show the mainland and marine habitat data for City. The Data-set is created using WorldView 2 satellite imagery acquired between 2012-2014 as part of a habitat mapping project. This satellite mission monitors the environment and takes images of the Earth and ocean. The habitat map is very accurate and reports 54 different colors, which represent 54 different features (*original features*) on both marine and terrestrial habitats. Each color in the map shown in Figure 2.3 corresponds to a starting feature.

The 54 starting features are grouped into three groups:

- *Seabed*. In this group, we include all the original features that can be traversed but that do not need to be investigated. Features of this are generically labeled as "seabed".
- *Mainland*. This group consists of all the features that represent terrain that can not be traversed by marine AUV, like for example airports, industry, farmland, road, etc. These features are therefore labeled as "mainland".
- *Basic feature*. This group represents the part of the seabed we are interested in. We grouped the original features into three classes: rock, coral, and grass.

2.2.2 Map processing and graph definition

The map considered corresponds to a surface too large to be analyzed in its entirety. For this reason, we focus on 12 snapshots taken from the map. Each snapshot corresponds to an area providing an interesting combination of the three basic features considered.

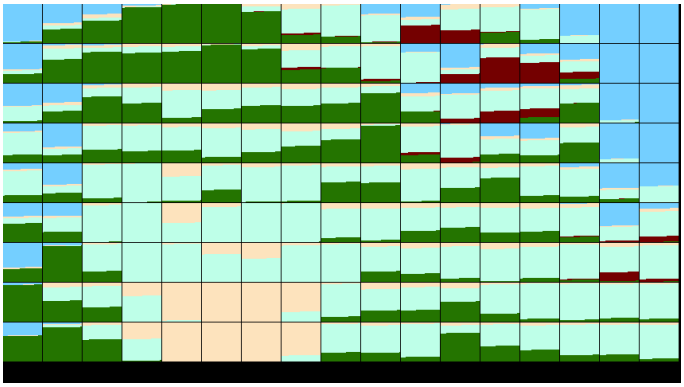


Figure 2.4: High granularity: squares of 200×200 m

The final benchmark used in the experiments is obtained after applying the following procedure to each of the 12 snapshots: we first apply to the snapshot a squared grid of fixed size that we will call it *granularity*. Secondly, in each square of the grid, we compute the percentage of the abundance of each feature, of the seabed and of the mainland.

In Figures 2.4, we show an example of how a snapshot is preprocessed according to the chosen granularity. In both figures, each square is filled with four colors: light blue (seabed), dark green (seagrass), light green (rock), red (coral), and peach (mainland). The height of each color is proportional to the percentage of the abundance of the corresponding feature. Starting from a snapshot and a chosen granularity, one instance that can be processed by the models described is created as follows:

- The graph $G = (N, A)$ is obtained by associating one node to each square.
- In principle it would be necessary to add a side for each pair of nodes distant a lower value equal to half of the range of each vehicle. However, this choice leads to an extremely dense graph. In our experiments, we have decided to limit ourselves to using a maximum of eight edges incident on each node: up, down, left, right, and the four on the diagonals. The values of $d_{i,j}$ are calculated as the Euclidean distance between the nodes in the graph.
- The profit p_i^f for feature f and node i represents the surface of the associated square covered by feature f. It is worth noting that a node (and the respective square) is considered fully visited. This is done because we assume that the distribution of the features is on average homogeneous on the full square and enough information is gathered by the AUV relative to that square.

2.3 Literature Analysis

The literature of exact [4] and heuristic [5, 6] algorithms for the VRP and its variant is vast, both in terms of applications and methodologies. For more details, we refer the reader to the following surveys: [7, 8, 9, 4, 10].

In [11] the authors propose a mixed integer linear program to plan the exploration of an area with a heterogeneous team of mobile agents. The communication is ensured by checking for each agent if it is within a certain distance of a list of other agents. In [12] a fleet of drones is used to investigate a set of points in an area (to each drone is assigned a single point). Once a drone visits the assigned point, it can be used as a relay in multi-hop communication. In [13] the multi-objective nature is related to different tasks, like monitoring, searching, and communication with the base station. The following works are quite similar to us: [14] and [15]. In [16] team of Micro-Aerial Vehicle used frontier-based exploration and coordination approach for exploration strategy. The goal of this paper is to reduce exploration time and energy consumption. They used an Ad-Hoc network. The difference from our work is that the system is weakly centralized. The leader is not predefined and can be changed during the mission.

In [17] multiple traveling salesmen (mTSP) paths are used to propose a positioning and trajectory planning algorithm for relay nodes to provide connectivity to drones that are out in the air.

In [18] a traveling salesman problem solved by a based partitioning algorithm is presented in this paper. UAVs need to repeatedly visit sensing locations while maintaining a multi-hop connection to the base station.

In [19] it is presented a simulation of a swarm of robots that keeps moving around in an enclosed environment and uses ad-hoc networking to communicate with each other.

3 Compact Model

In this chapter, we present a mathematical model used to simultaneously compute the optimal starting point and routes for a fleet of underwater AUVs, according to different optimization criteria. We will refer to this model as the compact model since we are using a polynomial number of variables and constraints, polynomial in the graph dimension.

3.1 Formulation

Parameters and variables In Table 3.1 and Table 3.2, we present the parameters and the decision variables used in the models presented in this section. The space is divided into a grid of squares. Each square is represented by a node. A graph G is used to represent the seabed, where each node i is associated with a given quantity p_i^f of feature f, representing the abundance of the feature f on that node (i.e. square). In our setting, we are considering the euclidean distance calculated in the generated graph. Thus, the value is not representing the real distance, but a normalized value that is equal to 1 when two nodes are adjacent. The maximum distance d_{max}^k can be different for every vehicle and it is normalized using the same ratio as the distance between two adjacent nodes. We define a variable $x_{i,j}^k$ for every arch and for every vehicle and each of them takes value 1 if the vehicle k pass through the arch connecting node i and j. Then, we have variables w_i that take value 1 if the base station is placed in node i. Variables y_i take value 1 if at least one of the vehicles collects the feature in that node. Lastly, variables t_i^k measure the arrival time at node i for vehicle k and take value 0 if the node is not

Table 3.1: Parameters - Compact Model

Name	Definition
$G = (N, A)$	Graph used for the routing of the AUVs
N	Set of nodes, $\|N\| = n_N$
A	Set of arcs, $\|A\| = n_A$
F	Set of features, $\|F\| = n_F$
K	Set of drones, $\|K\| = n_K$
$N^+(i)$	Set of nodes that are endpoints of outgoing arcs of node i
$N^-(i)$	Set of nodes that are endpoints of ingoing arcs of node i
p_i^f	quantity of feature f associated to node $i \in N$
$d_{i,j}$	Length of arc $\{i, j\} \in A$
d_{max}^k	Maximum distance associated to AUV $k \in K$

Table 3.2: Decision variables - Compact Model

Name	Definition	Range
$x_{i,j}^k$	Equal to 1 if AUV k goes from i to j	$\{0; 1\}$
w_i	Equal to 1 if node i is used as a starting point	$\{0; 1\}$
y_i	Equal to 1 if node i is visited by at least one AUV	$\{0; 1\}$
t_i^k	Arrival time at node i of AUV k	$\mathbf{R}_+$

visited by that vehicle. We need these variables to obtain feasible paths.

CONSTRAINTS The proposed mathematical models use the following sets of constraints:

$$y_i - \sum_{k=1}^{n_K} \sum_{j \in N^-(i)} x_{j,i}^k \leq 0 \qquad \forall i \in N \qquad (3.1)$$

$$\sum_{j \in N} w_i = 1 \qquad (3.2)$$

$$1 \geq \sum_{j \in N^-(i)} x_{j,i}^k \geq w_i \qquad \forall i \in N, k \in K \qquad (3.3)$$

$$1 \geq \sum_{j \in N^+(i)} x_{i,j}^k \geq w_i \qquad \forall i \in N, k \in K \qquad (3.4)$$

$$\sum_{j \in N^+(i)} x_{i,j}^k - \sum_{j \in N^-(i)} x_{j,i}^k = 0 \qquad \forall i \in N, k \in K \qquad (3.5)$$

$$\sum_{\{i,j\} \in A} d_{i,j} x_{i,j}^k \leq d_{max}^k \qquad \forall k \in K \qquad (3.6)$$

$$t_i^k + d_{i,j} - M(1 - x_{i,j}^k + w_j) \leq t_j^k \qquad \forall \{i,j\} \in A, k \in K \qquad (3.7)$$

$$t_i^k \leq M(1 - w_i) \qquad \forall i \in N, k \in K \qquad (3.8)$$

$$t_i^k \leq M \sum_{j \in N^-(i)} x_{j,i}^k \qquad \forall i \in N, k \in K \qquad (3.9)$$

Constraint (3.1) allows counting one node i as visited only if at least one of the arcs entering in i is equal to one. Constraint (3.2) imposes that one (and only one) node can be used as the base station. Constraints (3.3) and (3.4) ensure that the route of each AUVs starts and ends at the node selected as the base station. Constraint (3.5) impose that if an AUV visits node i it must also leave it. Constraint (3.6) imposes a limit on the maximum distance that each AUV can travel. Constraint (3.7) imposes the correct order of the nodes visited by vehicle k and avoids the generation of subtours. In our computation we used a value for the "big-M" constant equal to $M = d_{max} + \max_{(i,j) \in A} d_{i,j}$. Constraint (3.8) sets to zero the variable t associated with the base station. Constraint (3.9) sets to zero all the variables t associated with nodes that are not visited by any of the vehicles. If the starting position is fixed and if the objective function is linear, this problem reduces to the Team Orienteering Problem [1] or Vehicle Routing with profits [9, Chapter 10]. Many variants of the problem are present in the literature [8]. However, this is the first exact algorithm that includes the choice of the starting

point as part of the decision process. In Chapter 6 we show that the choice of the starting position is crucial for achieving good results in practice.

SINGLE FEATURE OBJECTIVE FUNCTION. In this work, we have n_F features and to each feature f we associate a set of profits p_i^f, with $f = 1, \ldots, n_F$ and $i = 1, \ldots, n_N$. To obtain a model able to find the set of paths that allow obtaining the maximum collected quantity of a given feature f, we use the following objective function:

$$\max \sum_{i=1}^{n_N} p_i^f y_i \ . \tag{3.10}$$

The objective function (3.10) is used to sum up the profits of feature f associated with each node.

3.2 MODEL IMPROVEMENT

REACHABILITY CONSTRAINTS The first set of inequalities allows having a tighter link between the y and the w variables. For each node $i \in N$, let $\mathcal{C}(i) \subseteq N$ be the set of nodes that are *reachable* from node i. A node i' is reachable from node i if there exists a path of length d_{max} that starts and ends in i that reaches node i'. With this definition, we define the following set of inequalities:

$$\sum_{j \in \mathcal{C}(i)} w_j \geq y_i \qquad \forall i \in N \ . \tag{3.11}$$

Inequalities (3.11) allows having a variable y_i with a value of one only if at least one of the nodes that can be reached by i is selected as starting position. Inequalities (3.11) are always valid (i.e., the model remains exact if we add them) and therefore in the following, we will consider them as always part of the model.

SYMMETRY BREAKING FORMULATION In order to reduce the feasible region, we want to get rid of that vehicles are not distinguishable from each other [20, 21]. For this reason, we need to restate the formulation of the problem. At first, we create a new set of variables as in Table 3.3. Then , we need to replace constraints (3.1) and (3.11) with the two following constraints:

Table 3.3: New variables y

Name	Definition	Range
y_i^k	Equal to 1 if node i is visited by AUV k	$\{0, 1\}$

$$y_i^k - \sum_{k=1}^{n_K} \sum_{j \in N^-(i)} x_{j,i}^k \leq 0 \qquad \forall i \in N, k \in K \tag{3.12}$$

$$\sum_{j \in \mathcal{C}(i)} w_j \geq \sum_{k \in K} y_i^k \qquad \forall i \in N \tag{3.13}$$

Lastly, we add a new constraint:

$$\sum_{i \in N} y_i^k p_i^f \geq \sum_{i \in N} y_i^{k-1} p_i^f \qquad \forall k \in \{2, ..., n_K\} \tag{3.14}$$

Constraints (3.14) introduce an order between vehicles. In fact, the total feature collected by each vehicle is decreasing with the number of vehicles. In this way, vehicles are no more exchangeable between each other and we discard solutions obtained by swapping only the order with which vehicles are chosen. It is noticeable that Constraints (3.13) assure that at most one vehicle retrieves the feature in every node.

Branching priority The solver we decided to use is CPLEX, which makes use of a branch and cut algorithm. Instead of leaving the solver free to choose in which variable it should branch, we set a level of priority for branching as follows:

- for the model with Constraints (3.1)-(3.9),(3.11), we set as high priority variables y_i. In this way, the model branches first on which nodes should be part of the solution.
- for the model with Constraints (3.2)-(3.9),(3.12)-(3.14), we defined a new set of variables:

$$z_i = \sum_{k \in n_K} y_i^k \qquad \forall i \in N \tag{3.15}$$

  Then, we set these new variables as high-priority variables for branching. In fact, variables z_i play the same role as variables y_i in the previous model.

4

Extended Formulation

In this chapter, we introduce a second model to solve our problem, solved by a column generation algorithm. This strategy differs from the one we already exploit since we consider a model where the variables represent all the feasible paths that our vehicle can travel. The number of variables considered is exponential in the size of the input, it is therefore impossible in practice to consider all these variables from the start, we initialize the model with a subset of paths and at each iteration, we try to add new variables in order to find the optimal solution.

4.1 Formulation

4.1.1 Master Problem

In order to apply the column generation algorithm, we reformulate our model as follows.

Parameters and variables. In Table 4.1 and Table 4.2 we present the parameters and the decision variables used in the models presented in this section. The setting is similar to the one presented in Chapter 3. We introduce a new set of variables, λ^p, where each of these variables takes value 1 if path p is one of the paths that are crossed by one vehicle.

Table 4.1: Parameters

Name	Definition
$G = (N, A)$	Graph used for the routing of the AUVs
N	Set of nodes, $\|N\| = n_N$
F	Set of features, $\|F\| = n_F$
K	Set of drones, $\|K\| = n_K$
P	Set of all feasible paths, $\|P\| = n_P$
p_i^f	quantity of feature f associated to node $i \in N$
y_i^p	indicates if node $i \in N$ is visited by path $p \in P$

Table 4.2: Decision variables

Name	Definition	Range
λ^p	Equal to 1 if path p is chosen by a AUV	$\{0; 1\}$
w_i	Equal to 1 if node i is used as a starting point	$\{0; 1\}$
y_i	Equal to 1 if node i is visited by at least one AUV	$\{0; 1\}$

CONSTRAINTS. The master model uses the following set of constraints:

$$y_i \leq \sum_{p \in P} \lambda^p y_i^p \qquad \forall i \in N \tag{4.1}$$

$$\sum_{p \in P} \lambda^p = n_K \tag{4.2}$$

$$\sum_{j \in N} w_i = 1 \tag{4.3}$$

$$w_i \leq \frac{1}{n_P} (\sum_{p \in P} \lambda^p y_i^p) \qquad \forall i \in N \tag{4.4}$$

$$\tag{4.5}$$

Constraint (4.1) allows counting one node i as visited only if at least one of the paths going through node i is chosen by a vehicle. Constraint (4.2) imposes that only n_K paths can be chosen. Constraints (4.3) imposes that one (and only one) node can be used as the base station. Constraint (4.4) imposes that one node i can not be the base station if there are less than n_K chosen paths going through node i.

SINGLE FEATURE OBJECTIVE FUNCTION. In this work, we have n_F features and to each feature f we associate a set of profits p_i^f, with $f = 1, \ldots, n_F$ and $i = 1, \ldots, n_N$. To obtain a model able to find the set of paths that allow obtaining the maximum collected quantity of a given feature f, we use the following objective function:

$$\max \sum_{i=1}^{n_N} p_i^f y_i \ . \tag{4.6}$$

The objective function (4.6) is used to sum up the profits of feature f associated to each node.

From now on, we refer to this problem as $MP(P)$, where P is the set of all possible paths. Since we can not work with all set of variables, but only with a subset of them, in the following, we are going to refer to $MP(\bar{P})$ as the master problem where the set of paths $\bar{P}$ is a subset of P, $\bar{P} \subset P$.

DUAL MODEL. In order to describe the procedure with which we are going to add new variables, we need to introduce the dual model of the $MP(\bar{P})$. We can summarize the dual problem as follow:

$$\min n_K \beta + \gamma \tag{4.7}$$

$$\alpha_i \geq p_i^f y_i \qquad \forall i \in N \tag{4.8}$$

$$\beta - \sum_{i \in p} (\alpha_i + \frac{\delta_i}{n_k}) \geq 0 \qquad \forall p \in \bar{P} \tag{4.9}$$

$$\gamma + \delta_i \geq 0 \qquad \forall i \in N \tag{4.10}$$

$$\alpha_i, \delta_i \geq 0 \qquad \forall i \in N \tag{4.11}$$

$$\beta, \gamma \in \mathcal{R} \tag{4.12}$$

Dual variables $\alpha_i, \beta, \gamma, \delta_i$ correspond to constraints (4.1) - (4.4) of the $MP(\bar{P})$ and dual constraints (4.8) - (4.10) are associated to variable y_i, λ^p, w_i of the primal problem.

4.1.2 Subproblem

As we explained in Chapter 1, to solve this particular formulation, we need an additional problem to generate paths. Regarding the pricing problem for VRP, we refer to the work of Desaulniers et al [22]. Among the all possible choice for the pricing subproblem, we decide to adapt a Resource Constrained Elementary Shortest Path Problem (RCESP).

Thanks to the work of Moshe Dror [23], it is shown that the RCESP is strongly NP-hard. To the best of our knowledge, the most used technique to solve the RCESP is dynamic programming. In this work, we decide to use an implementation of the algorithm proposed by Tilk et al. [24]. They extended the work of Righini and Salani [25] regarding the bounded bidirectional dynamic programming algorithm for RCESP. In particular, they exploit new strategies regarding asymmetric halfway points.

We can define the pricing problem as follows. Since we are tackling a maximization problem, we want to find new columns, specifically new paths, that have a positive reduced cost. The reduce cost of each column $\lambda_p \in P$ can be computed as:

$$\sum_{i \in p} (\alpha_i + \frac{\delta_i}{n_k}) - \beta > 0 \tag{4.13}$$

where the summation is done on all the nodes crossed by path p and the dual variables β, γ, δ_i correspond to constraints (4.2) - (4.4).

In order to find a new column that has a positive reduced cost, we create a graph where the arcs have non-negative weights that represents the euclidean distance d_{ij} between the linked nodes and non-positive weights on nodes with the following expression: $\alpha_i + \frac{\delta_i}{n_k}$. With this representation, the RCESP finds a non-elementary cycle that starts and ends in the base station, with a maximum viable distance equal to the maximum distance the vehicle can do and with a maximum reduced cost.

4.2 Algorithm

In this section we describe the algorithms we implemented to solve the extended formulation thanks to column generation. In particular, we show three different algorithms: a branch and price scheme and two heuristic diving algorithms.

4.2.1 Branch and Price

As we mentioned in Chapter 1, a branch and price algorithm is formulated as a combination of branch and bound with column generation. The algorithm starts with a column generation phase. When no more columns are added to the master problem, we have two possible cases. If the LP relaxation of the master problem has an integer solution, the algorithm terminates and the optimal solution is found. If the LP relaxation has a noninteger solution, the algorithm picks a variable with a fractional value and branches on that. For every branch, the algorithm first solves the LP relaxation, then a column generation phase is applied to add a new column and lastly, the LP relaxation is solved again with the new variables added. The branching scheme goes on as a usual branch and bound algorithm and terminates when no more branches are open.

There are still some problems to be addressed. First of all, we have to be careful to design a good branching scheme. In fact, adding constraints can modify the pricing and make it considerably harder to be solved in pratice. In our implementation, we choose to branch first on variables y_i and then on w_i since branching on these variables does not affect the pricing problem. Furthermore, dual constraints 4.9 are associated with variables λ^p meaning that only if we branch on variables λ^p there is a modification of the pricing problem. In principle to have an exact branching scheme, it would be necessary to implement another exact branching policy to avoid getting stuck with variables λ^p with rational values. In practice, given the size of the instances, we opted for a heuristic solution. Since it is possible that all variables y_i and w_i have

integer values after LP relaxation of the master problem, but not λ^p, we terminate the branch by subMIPing (i.e. by adding the integrality constraints).

Another choice we faced is how to perform branching. After several tries, we decide to branch on the variable with the value closer to 0.5.

The pseudocode of the algorithm is described in Algorithm 4.1.

Algorithm 4.1 Branch and Price

```
1:  Initialize P̄
2:  B = [Initial Master Problem]
3:  val_best_sol = −∞
4:  best_sol = []
5:  while B ≠ []
6:      Solve linear relaxation of MP(P̄)
7:      if Unfeasible
8:          Prune branch
9:      Solve subproblem - Column Generation Phase
10:     Solve linear relaxation of MP(P̄)
11:     if solution is integer
12:         if val_sol ≥ val_best_sol
13:             val_best_sol = val_sol
14:             best_sol = sol
15:             Prune Branch
16:     else if val_sol ≤ val_best_sol
17:         Prune Branche
18:     else
19:         Branch on variable
20:         Add branches to B
```

4.2.2 Model Improvement

Ordering of the base station Fixing a w variable is equivalent to deciding which node should be used a starting point (or base station).

During the branch and price algorithm, we explore all the possible base station in order to obtain an improving variable. We found out that the order with which we try all the base station is critical. We tested three different approaches:

- Following the order of the node

- Randomly try all the nodes
- Assigning at each node a probability value equal to the value of the corresponding w variable after solving the linear relaxation and, to not get stuck, adding a random component. In this way, the model selects first the base station that has a non-zero value in the last linear relaxation solved.

Sub-graph generation To solve the sub-problem through the dynamic programming algorithm we decide to use, it is necessary to create a graph from every base station with the respective dual weights. In generating this graph, we keep only the nodes that are reachable from the selected base station. By doing so, we can reduce the dimensionality of the problem and obtain a new path faster.

Multiple paths addition Thanks to the algorithm we chose for the resource constraint shortest path, we have the possibility to retrieve not only the best route but also a list of the best paths. In this way, we can add more than one path at each iteration, if the reduced cost is negative. This procedure could add more paths to the model, even useless ones, but the main advantage is that, on average, we get a good subset of paths faster.

4.3 Heuristic Approach

Since even with this new formulation, the problem could be hard to solve when the dimension of the graph is high, we develop some heuristic approach to find a feasible solution in a reasonable amount of time.

4.3.1 Diving Heuristic

One of the most effective depth-first heuristics search algorithms in the branch and price tree is a diving heuristic. At each node of the tree, the algorithm chooses a branch based on some greedy or rounding strategies. This fixing is different from the branching decision in the branch and price algorithm since we are not interested in exploring and balancing the tree, but we want to find a feasible solution faster. Furthermore, at each fixing, the LP relaxation of the master problem is solved again and new columns are generated, if it is possible. To do so, we have to be careful during the fixing phase, since we do not want to impair the column generation

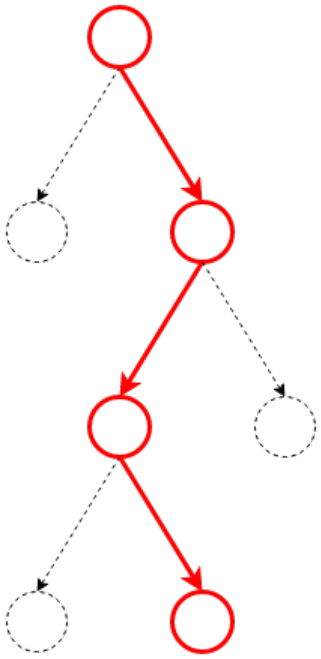


Figure 4.1: Main scheme for diving.

phase by adding some constraints that are incompatible with the pricing problem. A visual representation of the diving heuristic in a binary tree is shown in Picture 4.1.

In our implementation, we proceed as follows:

- Solve the LP relaxation of the master problem.
- Select a variable and fix its value to 1.
- Perform column generation after updating the master problem.
- Repeat the fixing T times and restart if the problem is shown as infeasible.
- SubMIPing to find a feasible solution after T fixing.

To avoid getting stuck to the same fixing scheme, we add the chosen node to a tabu-list and store the node for a couple of iterations. After that, we start by removing randomly nodes from the tabu list, to keep variability in the diving scheme. The pseudocode of the algorithm is described in Algorithm 4.2.

Algorithm 4.2 Diving scheme

```
1: Initialize $\bar{P}$
2: val_best_sol = $-\infty$
3: best_sol = []
4: Tabu_List = []
5: while $time \leq TL$
6:     while $k \leq MaxDepth$
7:         Solve linear relaxation of $MP(\bar{P})$
8:         if Unfeasible
9:             Prune branch
10:        Solve subproblem - Column Generation Phase
11:        Solve linear relaxation of $MP(\bar{P})$
12:        if solution is integer
13:            if val_sol $\geq$ val_best_sol
14:                val_best_sol = val_sol
15:                best_sol = sol
16:                Break
17:        else if val_sol $\leq$ val_best_sol
18:            Break
19:        else
20:            Choose a variable not in Tabu_List and fix the value
21:            Add chosen variable to Tabu_List
22:    Remove some elements from Tabu_List
```

4.3.2 Square diving

One of the most critical aspects of our formulation is the choice of the base station. In fact, if we keep the base station fixed, we obtain the formulation of a team-orienteering problem. To make the choice of the base station easier to tackle, we implement a heuristic diving where at each iteration we reduce the space of possible choice for the base station. The main idea is the following:

- After the first LP relaxation is solved and all the possible columns are generated, we look at the value of w_i variables and find the index i for which the value is higher. Then we center a square S_0 of side a_0 around this node.
- At the next iteration of the algorithm, the base station should be inside this new square.
- After solving the LP relaxation and generating new columns, we create a new square S_i with side a_i around the node with a higher w_i value and we reiterate this procedure until we have only a node inside this square.
- To find a feasible solution, we apply subMIPing.

To ensure convergence, the values of a_i is a decreasing sequence. In our particular implementation, we have the sequence as $12, 8, 4, 2, 1$. A graphical representation of this idea is shown in Picture 4.2.

Since we want to apply this procedure several times in a row, we do not want to end up always in the same base station. To avoid this phenomenon, we add a Tabu-list. The idea is that after a base station is chosen by the diving algorithm, it enters a Tabu-list and it cannot be chosen again for the next k iteration. In this way, we can explore different regions and generate more paths.

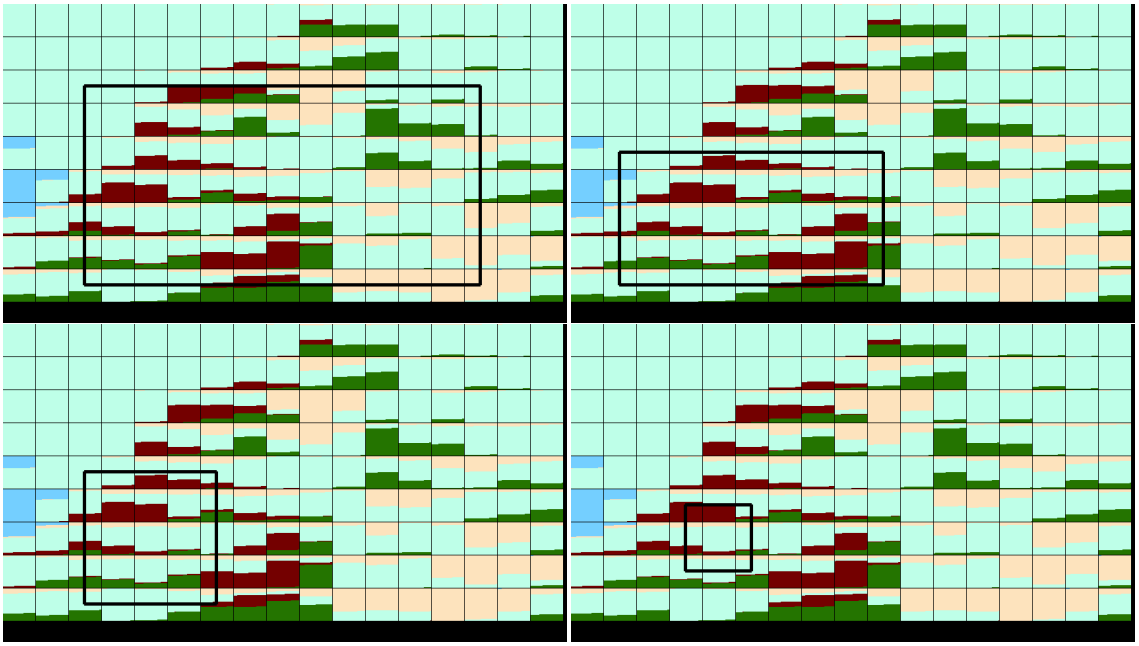


Figure 4.2: An iteration of diving with tabu-square.

5 Model with communication

Working in an underwater environment could be challenging, and we should expect possible malfunctioning of the vehicles. Retrieving a UAV without the support of locating systems like GPS could be a hard task. To avoid this scenario, we want to enhance our model ensuring that the vehicles can communicate with each other. The communication should happen to take into account the range of underwater transmission channels.

For the sake of simplicity, we want to impose that the vehicles should have a link with the base station when they still have half of their battery life. Since it is reasonable to think that when they have half of their autonomy missing they are starting to go back towards the base station. To do so, we impose a checkpoint half of the way and we verify that all the vehicles are communicating with a multi-hop link with the base station. The approach we are going to propose can be extended to a larger number of checkpoints, depending on the communication required by the user.

5.1 Formulation

Since we were able to find good results with a column generator algorithm, we propose a similar approach for our model with communication. We decide to implement the model in the same fashion, but we need to do different assumptions to keep the model feasible. At first, in this scenario, we take the position of the base station for granted. Then we choose to deal with the communication as follows. We ask for multi-hop communication. In this way, a vehicle could

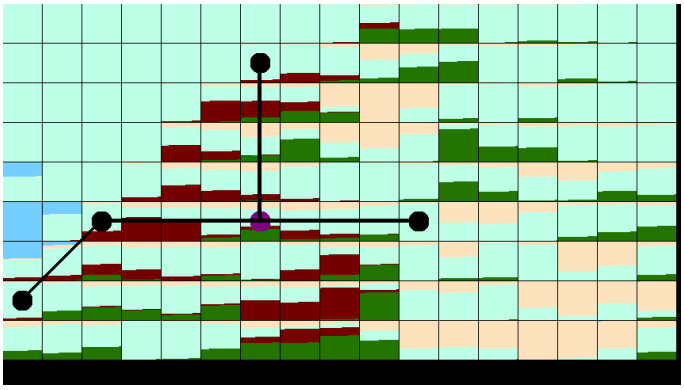


Figure 5.1: Multi-hop communication.

not be communicating directly with the base station, but it can be linked with another vehicle that is in range of the base station. Furthermore, we check if all the vehicles are communicating only when they reach half of their autonomy. Thanks to these assumptions, we are ready to define our model.

5.1.1 Master

Parameter and Variables. As shown in Table 5.1, we have a new set of parameters representing all the possible communication trees that allow each vehicle to be connected with the base station (example of communication tree). To generate all these possible trees, we adapt an algorithm developed by Karakashian, Choueiry, and Hartke [26].

We need to add another set of parameters to ensure that the vehicles are at the checkpoint at the required time interval. To do so, when we generate a path, we save as a parameter the midpoint of the path. Thus, if the model chooses that path, we know that the vehicle should be in that specific position during the communication checkpoint. In this way, it is easy to check if all the vehicles respect a particular communication tree.

Lastly, we need to handle the choice of the communication tree. We add a new set of variables, one for each tree. Each of these variables takes value 1 if the corresponding communication tree is chosen, 0 otherwise.

Constraints and Objective function. Since the aim of the model is the same as in the previous chapter, we take as the objective function the maximum over the chosen feature

Table 5.1: Parameters

Name	Definition
$G = (N, A)$	Graph used for the routing of the AUVs
N	Set of nodes, $\|N\| = n_N$
F	Set of features, $\|F\| = n_F$
K	Set of drones, $\|K\| = n_K$
P	Set of all feasible paths, $\|P\| = n_P$
T	Set of all communication tree, $\|P\| = n_T$
p_i^f	quantity of feature f associated to node $i \in N$
y_i^p	indicates if node $i \in N$ is visited by path $p \in P$
m_i^p	indicates if node $i \in N$ is midpoint of path $p \in P$
h_i^t	indicates if node $i \in N$ is part of communication tree $t \in T$

Table 5.2: Decision variables

Name	Definition	Range
λ^p	Equal to 1 if path p is chosen by a AUV	$\{0; 1\}$
y_i	Equal to 1 if node i is visited by at least one AUV	$\{0; 1\}$
c_t	Equal to 1 if tree t is used as communication tree	$\{0; 1\}$

collected by the fleet. The master model uses the following set of constraints:

$$\max \sum_{i=1}^{n_N} p_i^f y_i \tag{5.1}$$

$$[\alpha_i]\ y_i - \sum_{p \in P} y_i^p \lambda^p \leq 0 \qquad \forall i \in N \tag{5.2}$$

$$[\beta] \sum_{p \in P} \lambda^p = n_K \tag{5.3}$$

$$[\theta] \sum_{t \in T} c_t = 1 \tag{5.4}$$

$$[\mu_i] \sum_{p \in P} \lambda^p m_i^p - \sum_{t \in T} h_i^t c_t \leq 0 \qquad \forall i \in N \tag{5.5}$$

$$y, l, c \geq 0 \tag{5.6}$$

Constraint (5.2) allows counting one node i as visited only if at least one of the paths going through node i is chosen by a vehicle. Constraint (5.3) imposes that only n_K paths can be chosen. Constraints (5.4) impose that one (and only one) communication tree can be used. Constraints (5.5) impose that there is communication.

5.1.2 Subproblem

Dual model. In order to describe the procedure with which we are going to add new variables, we need to introduce the dual model of the $MP(\bar{P})$. We can summarize the dual problem as follow:

$$\min\ n_k \beta + \gamma + \theta \tag{5.7}$$

$$[y_i]\ \alpha_i \geq p_i^f \qquad \forall i \in N \tag{5.8}$$

$$[\lambda^p]\ -\sum_{i \in N} y_i^p \alpha_i + \beta + \sum_{i \in N} m_i^p \mu_i \geq 0 \qquad \forall p \in P \tag{5.9}$$

$$[c_t]\ \theta - \sum_{i \in N} h_i^t \mu_i \geq 0 \qquad \forall t \in T \tag{5.10}$$

Dual variables $\alpha_i, \beta, \theta, \mu_i$ correspond to constraints (5.2) - (5.5) of the $MP(\bar{P})$ and dual constraints (5.8) - (5.10) are associated to variable y_i, λ^p, c_t of the primal problem.

We can define the pricing problem as follows. Since we are tackling a maximization problem, we want to find new columns, specifically new paths, that have a positive reduced cost. The reduce cost of each column $\lambda_p \in P$ can be computed as:

$$\sum_{i \in N} y_i^p \alpha_i - \beta - \sum_{i \in N} m_i^p \mu_i \geq 0 \tag{5.11}$$

where the summation is done on all the nodes crossed by path p.

In order to find a new column that has a positive reduced cost, we create a graph where the arcs have non-negative weights that represents the euclidean distance d_{ij} between the linked nodes and non-positive weights on nodes with the following expression: $\alpha_i + \frac{\delta_i}{n_k}$. In this case, implementing a direct approach with RCESP is complex since we have to know which is the midpoint of the route. To avoid any complications, we decide to impose in advance the midpoint and find two shortest paths: one from the base station to the midpoint and another for coming back. When the first one is chosen, we avoid using the same arches by removing them from the graph.

5.1.3 Tree generation

The formulation we just presented requires the generation of all trees in the graph that contains the base station and 4 nodes, one for each vehicle. We refer to the algorithm developed by (add ref). This algorithm is able to find all trees in a given graph. Since we need only the trees that contain the base station, we modified the algorithm to obtain the trees that have that particular node.

Thanks to this technique, we retrieve all the trees. This number grows combinatorically with the dimension of the input. In our specific application, with a graph of dimension 17×9, the number of subtrees with 5 nodes is bounded by $\simeq 100000000$ (found experimentally). If we add this amount of variables to our model, it will become really hard to solve in a reasonable amount of time. To speed up the solution, we heuristically choose a subset of trees. We decide to keep only the trees where all nodes are far from the base station of at least $d_{comm} - \delta$. This choice is due to the fact that we impose a checkpoint at half-time of the mission and we expect the vehicle to be the furthest possible. To reduce further the number of trees, we decide to keep only the trees where at least two nodes are further than d_{comm}. The reason behind this choice is that we expect that the vehicles are exploring different regions of the graph.

Thanks to this observation, we were able to reduce the number of trees to roughly 10000.

5.1.4 Branch and Price

The branch and price algorithm developed for this model is similar to the one we presented in Chapter 4. The scheme is exactly the same for most of the operations, but there are a couple of changes. At first, we do not have w_i variables, but we have the new c_t variables. So, the branching is done first on the y_i as before, and then on c_t. Both of this fixing does not interfere with the pricing problem. Since the pricing problem has been modified, to add new columns we have to explore all the possible midpoints. This idea is similar to what we did for the base station in the first branch and price algorithm we presented. In this case, we try to add a new column by choosing every midpoint possible. If it is not possible to add any new variables, the pricing problem ends.

6 Results

The aim of our project is to find a model able to perform better than the compact model describe in Chapter 4. Taking into account the setting we described in Chapter 2, our fleet can collect three different types of features: coral, grass, and rock. Since we worked with a single feature objective function, we present the results for every feature. Furthermore, all the fine-tuning of the model was carried on for the coral feature, since it is the more challenging to collect. This is due to the fact that the coral is sparse and present in less quantity with respect to grass and rock.

In order to compare the performance, we define a measure called GAP:

$$GAP(method) = \frac{UB - LB(method)}{UB} * 100 \tag{6.1}$$

where UB is the upper bound of the compact model and $LB(method)$ stands for the Lower Bound of the selected method.

First, we describe the setting in which the experiment was carried on:

- We take into account a fleet made by 4 vehicles
- Each vehicle has an autonomy of 16.
- The graph is the one described in Subsection 2.2.2.
- The time limit for each model tested is 2 hours.

To have a better understanding, we recollect the model and their specification.

- **Compact model**: it is the model described in Chapter 3, where all the model improvement described are adopted.
- **Branch and price**: the model is described in Section 4.2.1.
- **Diving heuristic** it is the first heuristic algorithm in Section 4.3.1
- **Square diving**: heuristic algorithm presented in Section 4.3.2

6.1 Coral

We start by presenting the results obtained when the objective of the mission is to collect coral. In the following, we describe all the values that we measured during the experiment:

- **LB**: it is the lower bound found by the model. It is the value of the best integer solution found before the time limit is reached and is the amount of feature that has been collected.
- **UB**: it is the best upper bound of the total amount of feature that can be collected found by CPLEX during the execution of the solver.
- **Time**: it is the time in seconds. When the value is TL, it means that it has reached the given time limit.
- **BS**: they are the euclidean coordinate in the graph of the base station.

In Table 6.1 and 6.2 we present the results for our four algorithms. We highlight in bold the best results for each image in both tables.

As we expected, our column generation formulation performs better than the compact model. Furthermore, it is possible to notice how the choice of the base station is a crucial aspect to find a better solution. In Table 6.3 we compute the average distance of the base station between the models. As we can see, the average distance of the base station between the compact model and the others is higher and this could be a crucial aspect to obtain the maximum number of feature collected.

To have a better comparison of the model, we calculate the GAP for each method. In Table 6.4 we present the values of GAP for the methods we tested.

Table 6.1: Coral: results for exact models.

	Compact Model				Branch and Price		
Image	LB	UB	Time	BS	LB	Time	BS
1	0.4192	0.5452	TL	(5,5)	**0.5168**	TL	(10,5)
2	9.6504	14.8969	TL	(12,1)	**10.9532**	TL	(10,2)
3	3.1876	4.7076	TL	(9,3)	**4.2876**	TL	(6,6)
4	**3.9348**	3.9348	346	(12,6)	**3.9348**	8	(13,6)
5	9.844	9.8496	TL	(5,3)	**9.8488**	TL	(4,4)
6	5.1612	5.46	TL	(7,4)	**5.4564**	TL	(7,6)
7	7.7696	11.6725	TL	(11,5)	**7.7732**	TL	(11,4)
8	7.574	13.7625	TL	(3,2)	**9.754**	TL	(9,3)
9	3.6108	5.3468	TL	(2,0)	**3.8604**	TL	(4,4)
10	3.6296	5.8319	TL	(13,4)	**3.972**	TL	(12,3)
11	**3.8284**	3.8284	14	(11,6)	**3.8284**	9	(12,6)
12	3.7428	4.1488	TL	(2,3)	**3.7804**	TL	(4,3)

Table 6.2: Coral: results for heuristic algorithm.

	Diving on y			Diving on w (square)		
Image	LB	Time	BS	LB	Time	BS
1	**0.5136**	TL	(10,5)	0.4768	TL	(10,4)
2	10.9532	TL	(10,1)	**11.6316**	TL	(11,2)
3	**4.2876**	TL	(6,6)	4.2872	TL	(6,4)
4	**3.9348**	8	(13,6)	**3.9348**	8	(13,6)
5	**9.838**	TL	(6,3)	9.7868	TL	(6,3)
6	**5.4588**	TL	(7,6)	**5.4588**	TL	(7,6)
7	8.2068	TL	(9,4)	**8.3896**	TL	(9,3)
8	**9.6012**	TL	(11,1)	9.3604	TL	(7,3)
9	4.1052	TL	(7,2)	**4.1696**	TL	(7,2)
10	**4.0000**	TL	(13,2)	3.9372	TL	(12,3)
11	**3.8284**	9	(12,6)	**3.8284**	9	(12,6)
12	**3.7804**	TL	(4,3)	3.7736	TL	(4,4)

Table 6.3: Coral: Average distance of the base station.

	Root Node	Branch and Price	Diving on y	Square diving
Root Node	0.0000	2.6552	2.9938	2.5552
Branch and Price	2.6552	0.0000	1.0904	1.2565
Diving on y	2.9938	1.0904	0.0000	1.0250
Square diving	2.5552	1.2565	1.0250	0.0000

Table 6.4: GAP value for coral.

Model	GAP
Compact Model	21.21
Branch and Price	14.31
Diving on y	**13.73**
Square diving	13.97

6.2 Grass

In Table 6.1 and 6.2 we present the results for our four algorithms. We highlight in bold the best results for each image in both tables.

Table 6.5: Grass: results for exact models.

	Compact Model				Branch and Price		
Image	LB	UB	Time	BS	LB	Time	BS
1	6.6128	7.3520	TL	(9,5)	**6.8180**	TL	(9,5)
2	5.9916	6.4696	TL	(6,3)	**6.4456**	TL	(4,3)
3	24.3416	27.7247	TL	(10,3)	**24.5076**	TL	(12,2)
4	21.2356	30.2898	TL	(8,3)	**23.8416**	TL	(4,5)
5	10.2880	13.4080	TL	(5,8)	**11.1636**	TL	(10,4)
6	17.9652	24.2356	TL	(3,1)	**19.1600**	TL	(4,1)
7	**30.8944**	33.8258	TL	(9,4)	30.3224	TL	(7,3)
8	**23.2376**	25.3404	TL	(10,3)	22.5128	TL	(10,2)
9	6.7520	12.8202	TL	(4,2)	**10.5260**	TL	(11,2)
10	3.4080	4.0992	TL	(12,3)	**3.7028**	TL	(9,0)
11	22.2164	28.2381	TL	(8,5)	**22.3696**	TL	(10,5)
12	3.9652	4.4412	TL	(3,3)	**3.9696**	TL	(4,1)

As we did for the coral, we show how the choice of the base station plays a crucial role to carry on a successful mission. In Table 6.7 we compute the average distance of the base station between the models. As we can see, the average distance of the base station between the com-

Table 6.6: Grass: results for heuristic algorithm.

	Diving on y			Diving on w (square)		
Image	LB	Time	BS	LB	Time	BS
1	6.6608	TL	(9,5)	**6.8168**	TL	(9,5)
2	**6.4456**	TL	(4,4)	**6.4456**	TL	(4,5)
3	**24.5700**	TL	(11,2)	24.4320	TL	(10,3)
4	**22.5316**	TL	(3,6)	22.2960	TL	(3,5)
5	**11.1636**	TL	(10,4)	**11.1636**	TL	(10,4)
6	18.9828	TL	(4,1)	**19.2056**	TL	(4,1)
7	**29.8112**	TL	(7,3)	29.6332	TL	(7,3)
8	23.2108	TL	(10,3)	**23.3168**	TL	(10,3)
9	**10.2032**	TL	(12,2)	**10.2032**	TL	(12,2)
10	**3.6968**	TL	(9,0)	3.6712	TL	(9,0)
11	22.1744	TL	(5,4)	**22.4276**	TL	(5,4)
12	**3.9696**	TL	(4,3)	**3.9696**	TL	(3,1)

pact model and the others is higher and this could be a crucial aspect to obtain the maximum number of feature collected.

Table 6.7: Grass: Average distance of the base station.

	Root Node	Branch and Price	Diving on y	Square diving
Root Node	0.0000	2.3686	2.2106	2.3088
Branch and Price	2.3686	0.0000	1.0428	1.1113
Diving on y	2.2106	1.0428	0.0000	0.4709
Square diving	2.3088	1.1113	0.4709	0.0000

To have a better comparison of the model, we calculate the GAP for each method. In Table 6.8 we present the values of GAP for the methods we tested.

Table 6.8: GAP value for grass.

Model	GAP
Compact Model	18.49
Branch and Price	**13.22**
Diving on y	13.98
Square diving	13.82

6.3 Rock

In Table 6.1 and 6.2 we present the results for our four algorithm. We highlight in bold the best results for each image in both tables.

Table 6.9: Rock: results for exact models.

	Compact Model				Branch and Price		
Image	LB	UB	Time	BS	LB	Time	BS
1	8.6068	16.6296	TL	(5,14)	**14.914**	TL	(4,10)
2	**42.496**	49.8497	TL	(6,5)	38.6032	TL	(5,3)
3	**36.178**	51.7415	TL	(3,5)	34.8908	TL	(3,4)
4	32.986	46.5455	TL	(5,9)	**36.0952**	TL	(5,9)
5	35.5696	54.4654	TL	(3,15)	**44.1148**	TL	(3,3)
6	**36.9376**	44.4408	TL	(3,6)	33.9116	TL	(3,4)
7	39.2816	51.2107	TL	(4,16)	**39.8048**	TL	(2,15)
8	35.6124	55.3081	TL	(2,8)	**43.1684**	TL	(4,5)
9	26.0236	41.2702	TL	(2,9)	**29.5876**	TL	(2,11)
10	27.4584	39.8959	TL	(4,13)	**27.8688**	TL	(5,11)
11	25.7488	35.347	TL	(3,9)	**27.5424**	TL	(1,8)
12	28.6748	43.0733	TL	(6,2)	**31.8144**	TL	(7,9)

Table 6.10: Rock: results for heuristic algorithm.

	Diving on y			Diving on w (square)		
Image	LB	Time	BS	LB	Time	BS
1	**14.8916**	TL	(4,10)	14.8152	TL	(4,10)
2	**38.5332**	TL	(3,4)	37.1568	TL	(3,4)
3	**35.5856**	TL	(3,4)	**35.5856**	TL	(3,4)
4	34.51	TL	(5,12)	**34.81**	TL	(6,11)
5	**45.1148**	TL	(3,3)	43.2284	TL	(3,3)
6	**33.9428**	TL	(2,5)	33.5772	TL	(3,5)
7	**39.9588**	TL	(2,15)	39.8048	TL	(2,15)
8	42.6408	TL	(3,4)	**43.5472**	TL	(3,4)
9	**31.9156**	TL	(3,11)	29.2896	TL	(2,11)
10	**27.6976**	TL	(4,10)	27.292	TL	(3,12)
11	**26.436**	TL	(1,8)	26.1308	TL	(1,8)
12	29.854	TL	(7,9)	**29.8556**	TL	(7,4)

As we did for the coral, we show how the choice of the base station plays a crucial role to carry on a successful mission. In Table 6.11 we compute the average distance of the base sta-

tion between the models. As we can see, the average distance of the base station between the compact model and the others is higher and this could be a crucial aspect to obtain the maximum number of feature collected.

Table 6.11: Rock: Average distance of the base station.

	Root Node	Branch and Price	Diving on y	Square diving
Root Node	0.0000	3.3953	3.8002	3.1472
Branch and Price	3.3953	0.0000	0.8732	1.1769
Diving on y	3.8002	0.8732	0.0000	0.8875
Square diving	3.1472	1.1769	0.8875	0.0000

To have a better comparison of the model, we calculate the GAP for each method. In Table 6.12 we present the values of GAP for the methods we tested.

Table 6.12: GAP value for rock.

Model	GAP
Compact Model	30.12
Branch and Price	**23.45**
Diving on y	23.75
Square diving	24.90

6.4 Communication Results

As we mentioned before, it is important for our application to ensure stable communication. We decide to run our model trying to optimize the amount of feature collected. We choose as base station the one selected by the best model without communication. In this way, we can see our much feature collected we lose after imposing the communication constraint. The model we run is described in Chapter 5. The algorithm used is branch and price with a time limit of 2 hours.

Even if we do not impose that there should be some *relay* vehicles meaning that these vehicles are there just to ensure communication, we notice that in some cases there are vehicles that do not collect a large amount of feature, but they ensure that the communication is there. These vehicles explore the graph, but they go to a checkpoint to increase exploration for another vehicle at their expense. An example of this phenomenon can be seen in Figure 6.1. The blue

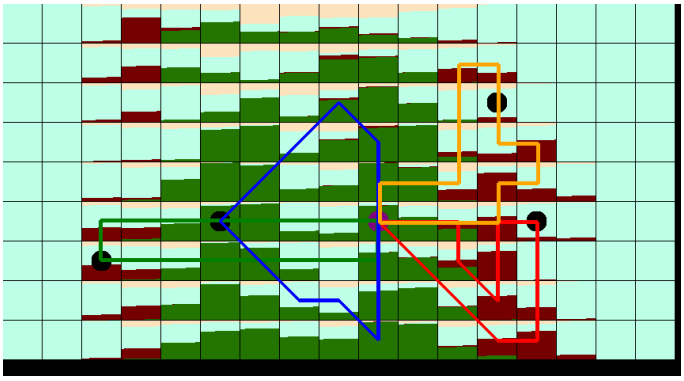


Figure 6.1: Example of relay vehicle.

vehicle (blue path) is basically not collecting any feature, but it allows the green vehicle (green path) to reach the feature on the bottom left part of the graph.

It is possible to see the results for all instances in Appendix A.

Here there are the results for the communication model for the three different features: coral, grass, and rock. In the next tables. the following quantities are reported:

- LB is the value of the best solution found by the model with communication.
- $\overline{\text{LB}}$ is the value of the best solution found by the model without communication.
- Time is the total amount of time needed by the model. If the value is TL, it means that after 2 hours the execution has been stopped.

In particular, looking at the results, it can be noticed that for some instances the best solution has a higher value than the one without communication. This phenomenon happens since once the base station is fixed, it is easier to find the right paths. It can be seen as an indicator of how difficult the problem is and of the fact that it is not trivial to optimize both the base station and the paths.

Table 6.13: Coral: experimental results with communication.

Image	LB	$\overline{\text{LB}}$	Time
1	0.5136	0.5168	70
2	12.1168	11.6316	TL
3	4.1724	4.2876	TL
4	3.9292	3.9348	150
5	9.5096	9.8488	TL
6	5.4560	5.4588	TL
7	7.9872	8.3896	TL
8	9.0512	9.6012	222
9	3.2612	4.1696	227
10	3.8696	4.0000	TL
11	3.8284	3.8284	282
12	3.7404	3.7804	TL

Table 6.14: Grass: experimental results with communication.

Image	LB	$\overline{\text{LB}}$	Time
1	6.5924	6.8180	905
2	6.3048	6.4456	TL
3	24.8492	24.4700	TL
4	23.7552	23.8416	TL
5	9.9728	11.1636	TL
6	18.7476	19.2056	TL
7	30.7580	30.8944	TL
8	23.3424	23.3168	TL
9	9.7312	10.5260	TL
10	3.5100	3.7028	TL
11	22.3612	22.4276	TL
12	3.8512	3.9696	218

Table 6.15: Rock: experimental results with communication.

Image	LB	$\overline{\text{LB}}$	Time
1	13.6456	14.9140	TL
2	44.3296	42.4960	TL
3	38.1036	36.1780	TL
4	36.6932	36.0952	TL
5	42.1324	45.1148	TL
6	37.3548	36.9376	TL
7	40.9576	39.9588	TL
8	46.1676	43.5472	TL
9	28.6768	31.9156	TL
10	28.6272	27.8688	TL
11	26.0728	27.5424	TL
12	34.9648	31.8144	TL

Conclusion

Nowadays, there is an increasing interest in using collaborative multi-vehicle systems for underwater exploration and monitoring, to achieve a more reliable and wider underwater area coverage. In fact, a fleet of AUVs equipped with cheaper sensors, since it can provide detailed information (images, videos) of larger areas, can achieve enhanced and more extended perception if compared to the one provided by a single, high-performant AUV.

The path planning of the mission plays a crucial role in energy saving and data retrieving. Since it is possible to find data that can give a hint on what there could be on the seabed, it is reasonable to design an a priori mission plan to achieve better perception. This kind of problem belongs to the large class of Vehicle Routing Problem and it is possible to define a model as an optimization problem.

In our work, the optimization problem has been exactly solved through several heuristic algorithms based on column generation. After developing an exact approach and given the complexity of the problem, we develop a heuristic branch and price algorithm. Starting from this idea, we designed two pure heuristic techniques based on the general scheme of *diving heuristics*. We experimentally showed that they all outperform the results obtained with a compact model.

The algorithms have been applied to real environmental data, specifically underwater maps based on satellite imagery provided by the Abu Dhabi Environmental Authority. An example of a use case is the collection of detailed information on the health and conservation status of corals, or different species of seagrass, as it is needed by local Environmental Authorities.

Furthermore, the model has been enhanced with communication constraints. This requirement is necessary since we are operating in an underwater environment, where it is difficult to retrieve a vehicle due to the absence of GPS.

As an outcome of this thesis, a paper has been published [27] and two are under preparation.

A

To have a general idea about what is going on in every instance, we show the results of the best solution without communication and with communication. The images are organized as follows: on the left part, there are the results without communication, and on the right side the results with communication. From top to bottom the images are divided by the feature collected: coral (red), grass (dark green), and rock (light green) as in Table A.1.

Coral	Coral with communication
Grass	Grass with communication
Rock	Rock with communication

Table A.1: Schematic explanation of the organization of the images.

In each figure, each path is denoted by a different color: green, red, yellow, and blue. The base station is represented by a purple dot. The black dot on the instances with communication represents the halfway point where we ensure that the vehicle is passing by for achieving communication.

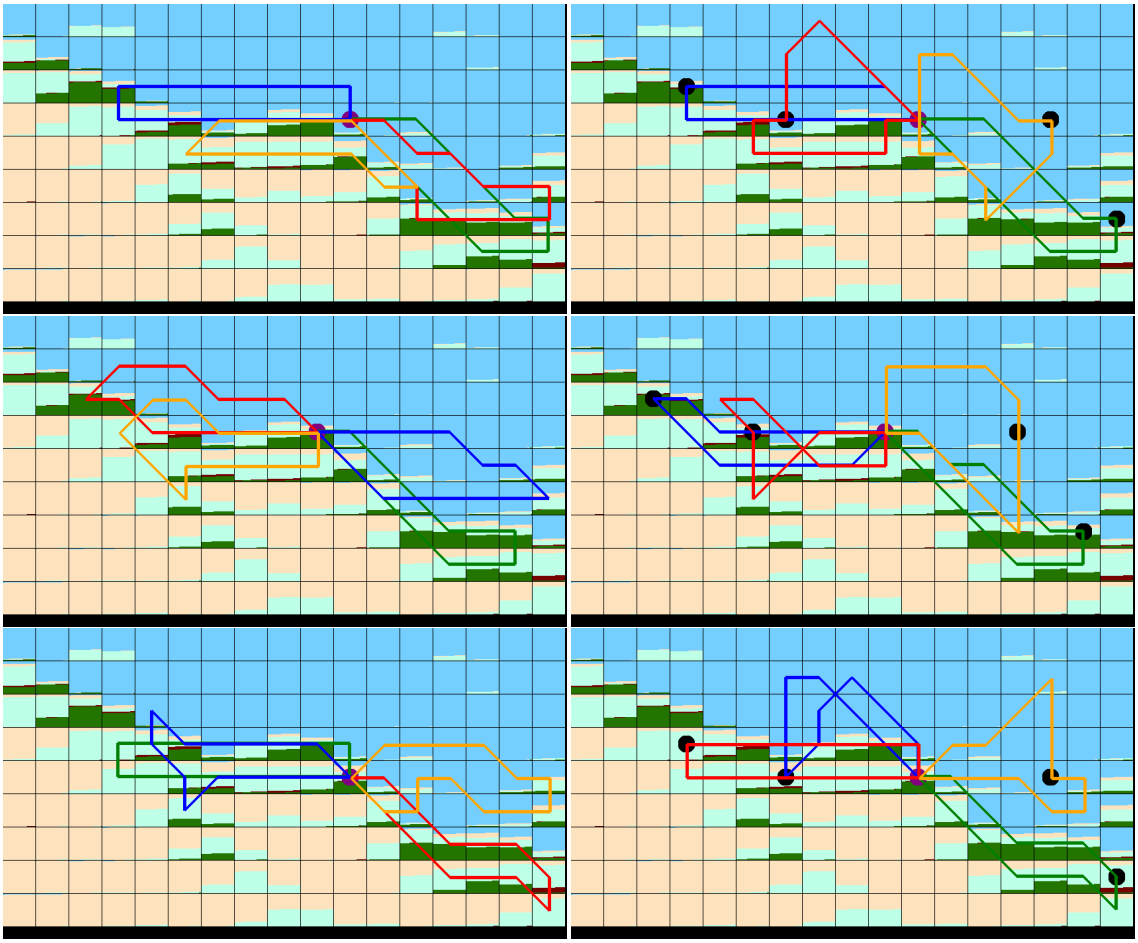


Figure A.1: Results on instance 1.

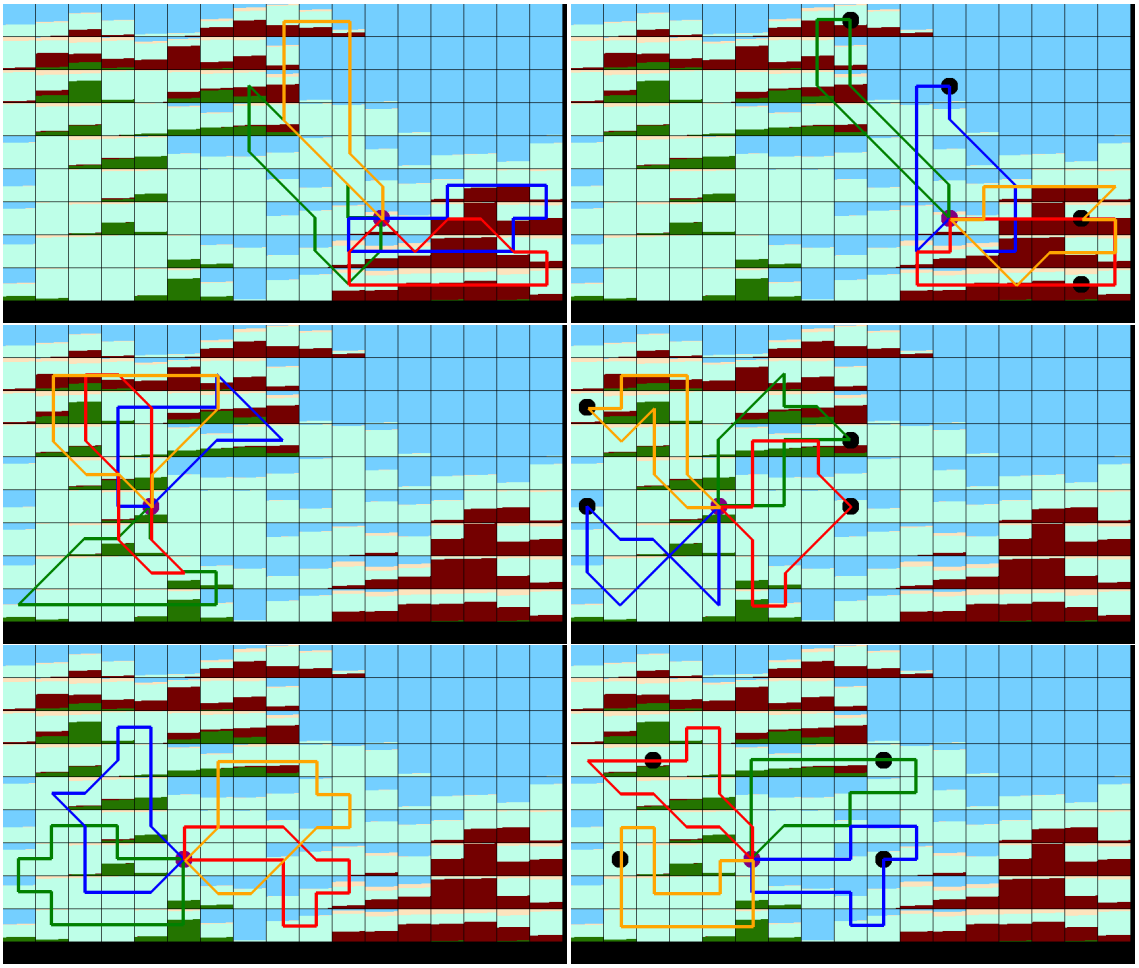


Figure A.2: Results on instance 2.

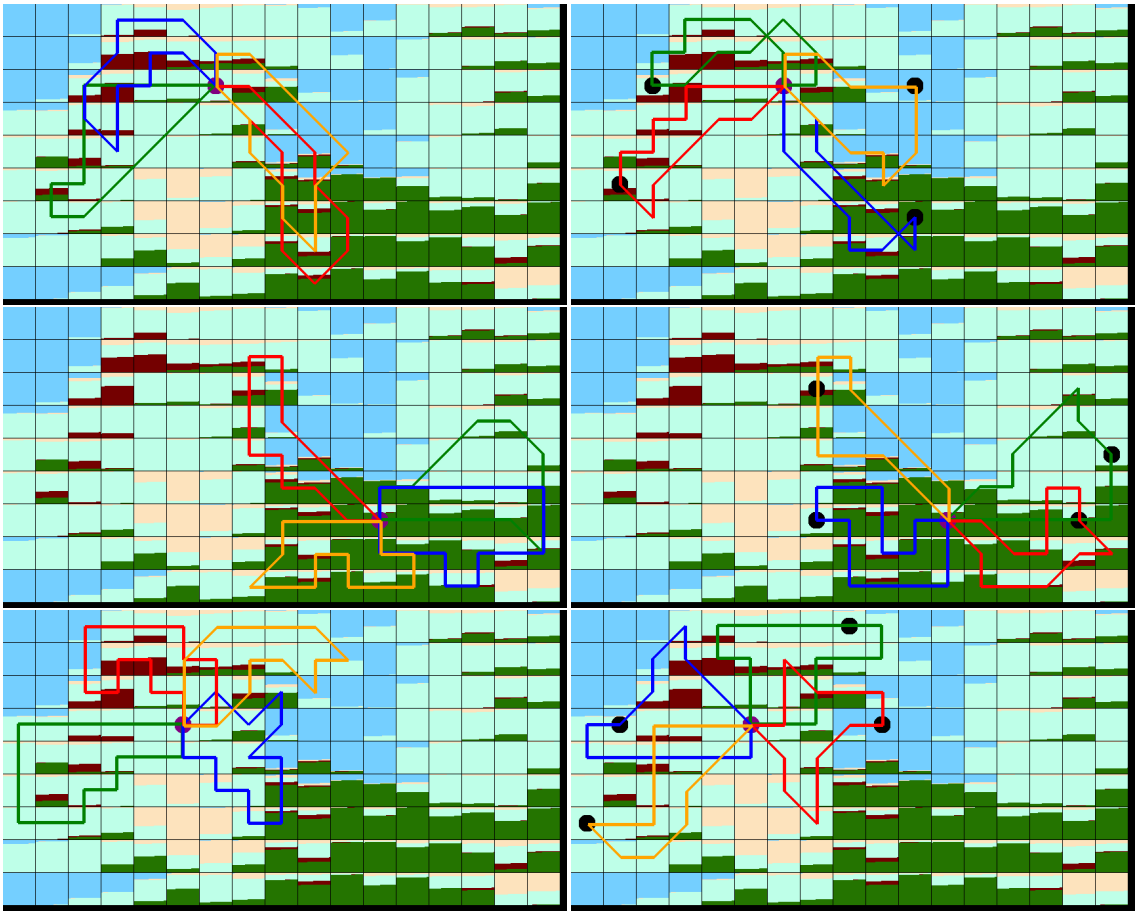


Figure A.3: Results on instance 3.

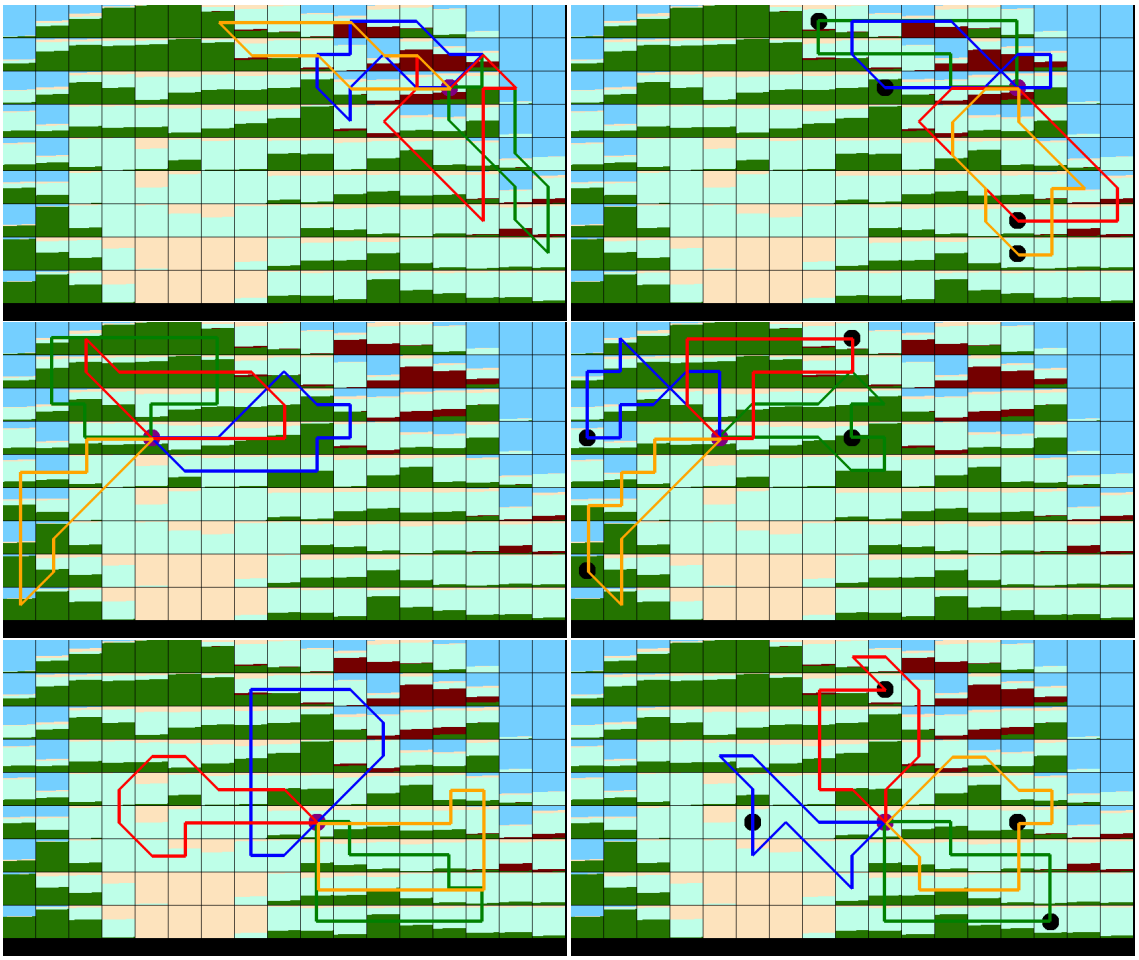


Figure A.4: Results on instance 4.

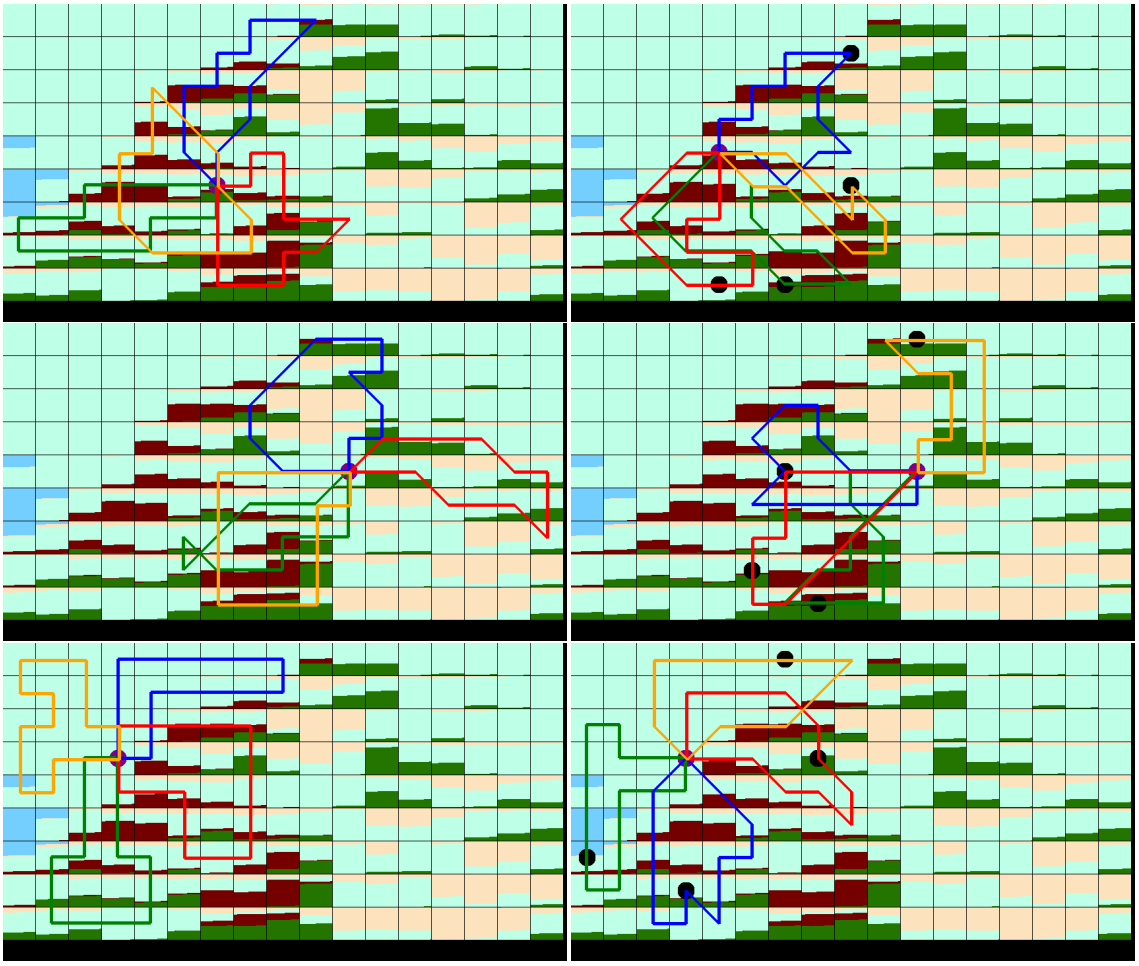


Figure A.5: Results on instance 5.

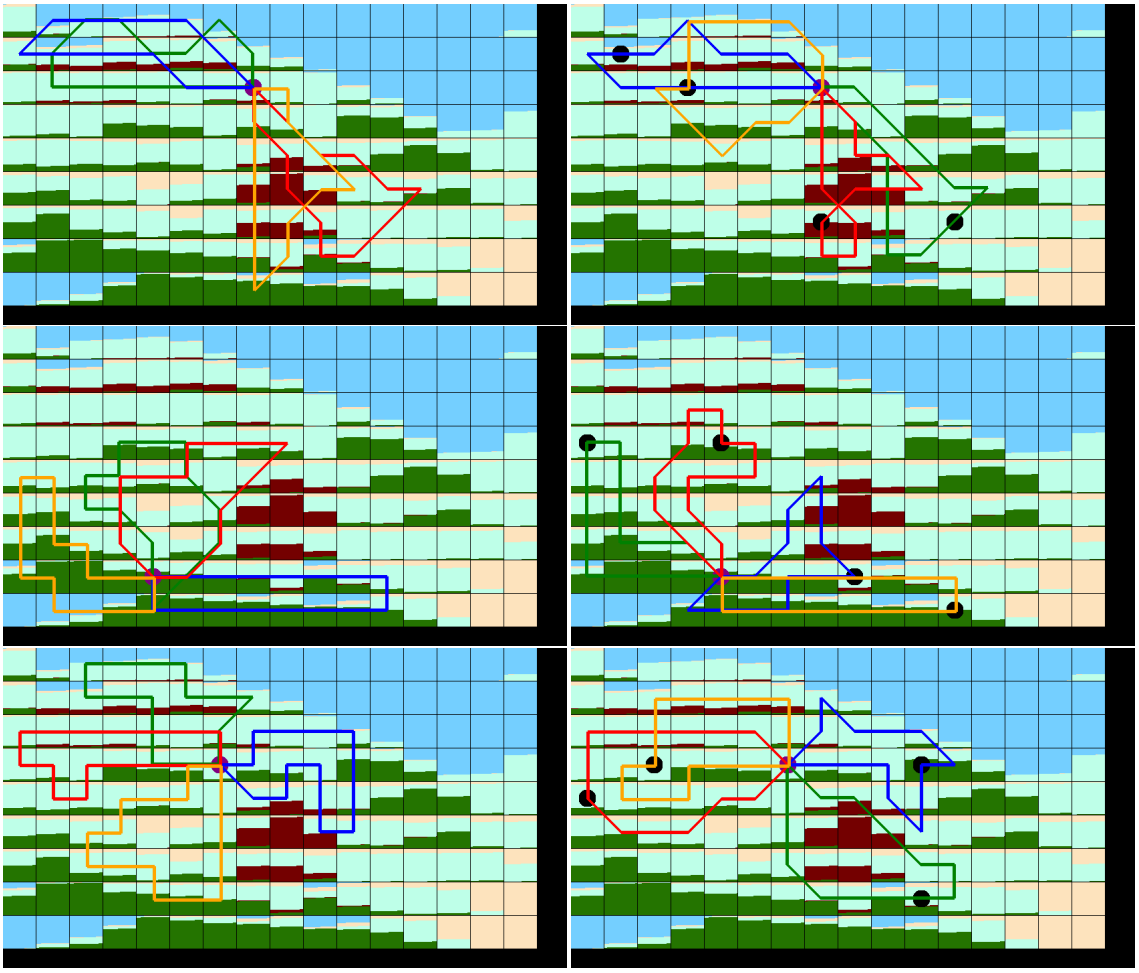


Figure A.6: Results on instance 6.

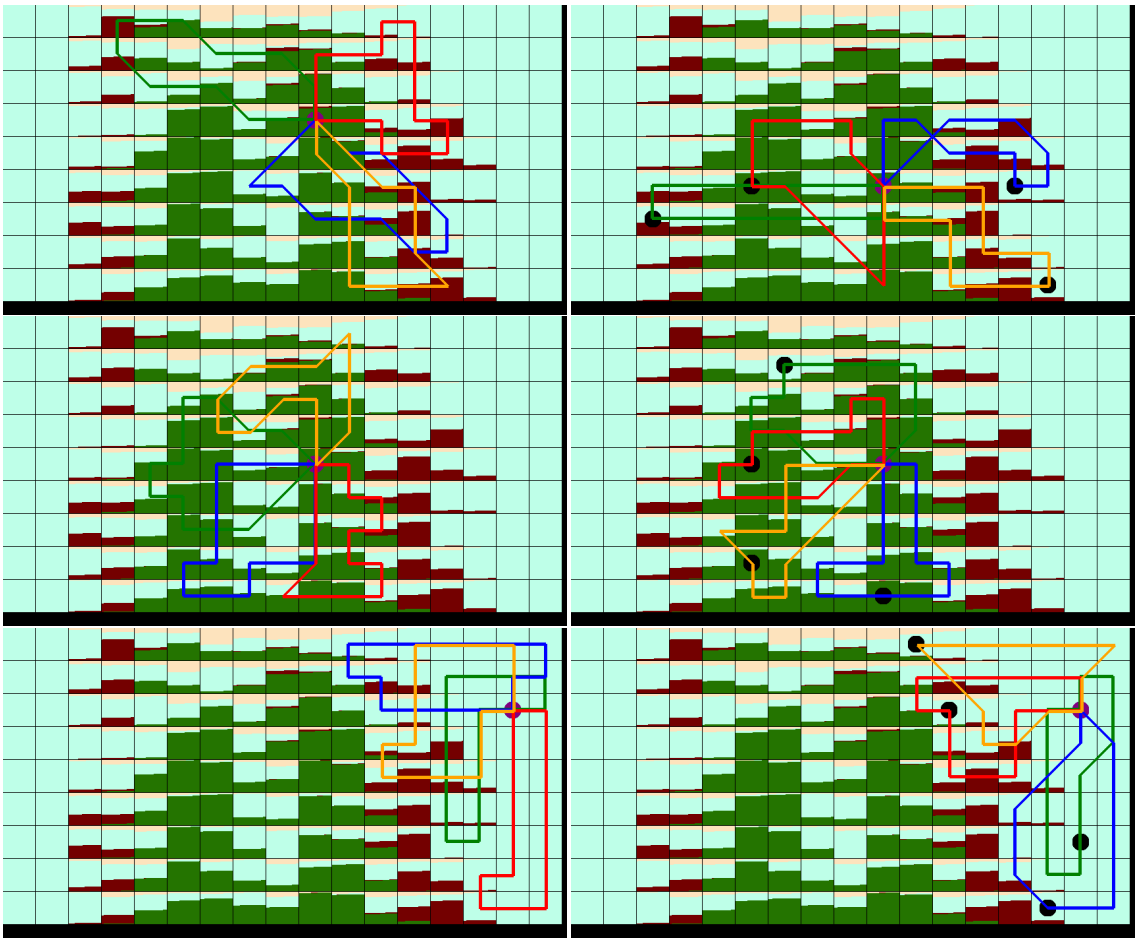


Figure A.7: Results on instance 7.

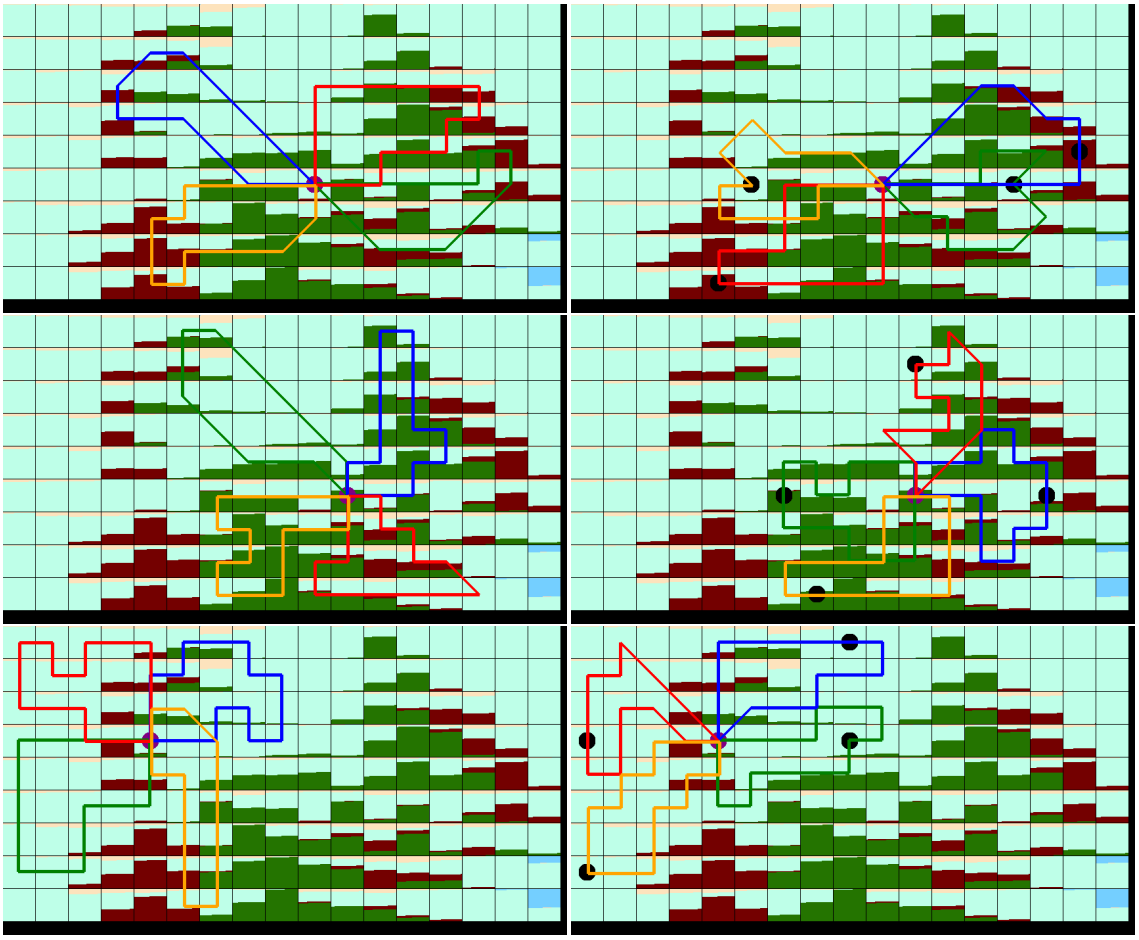


Figure A.8: Results on instance 8.

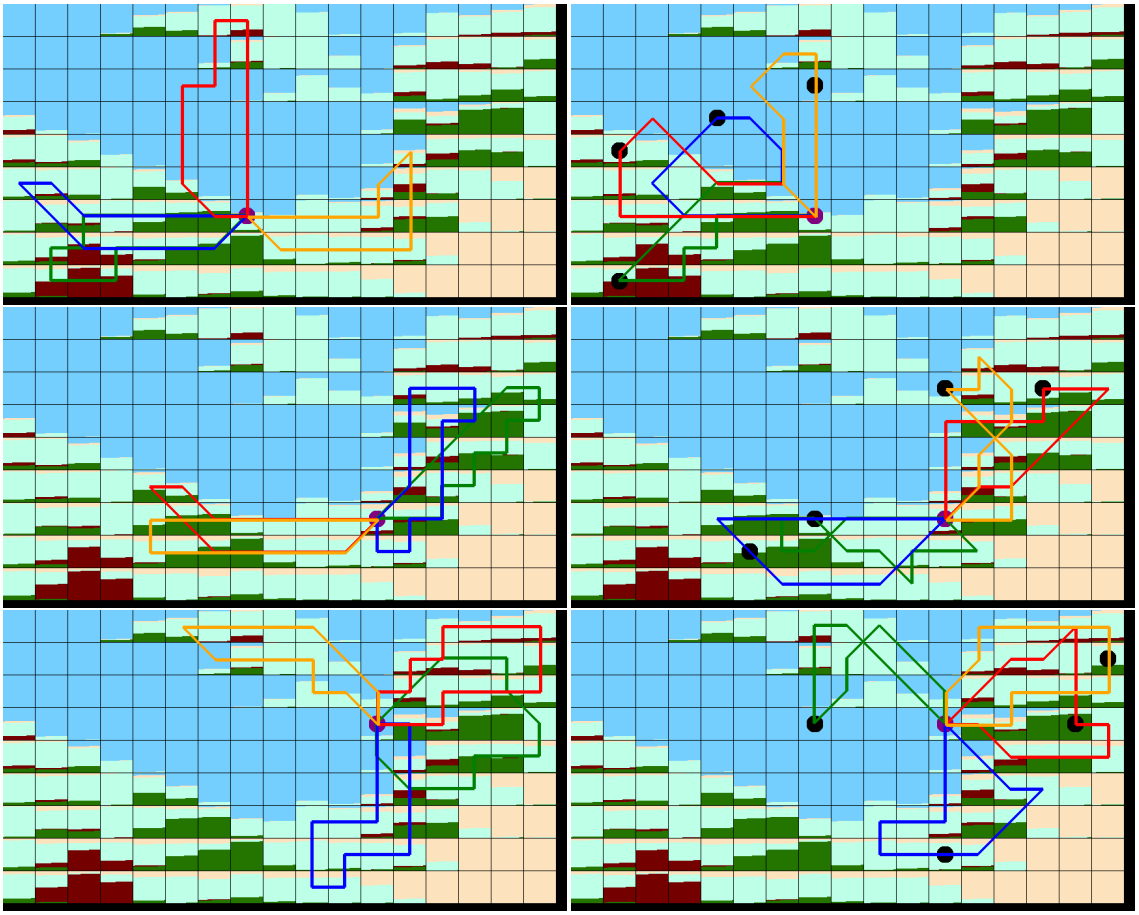


Figure A.9: Results on instance 9.

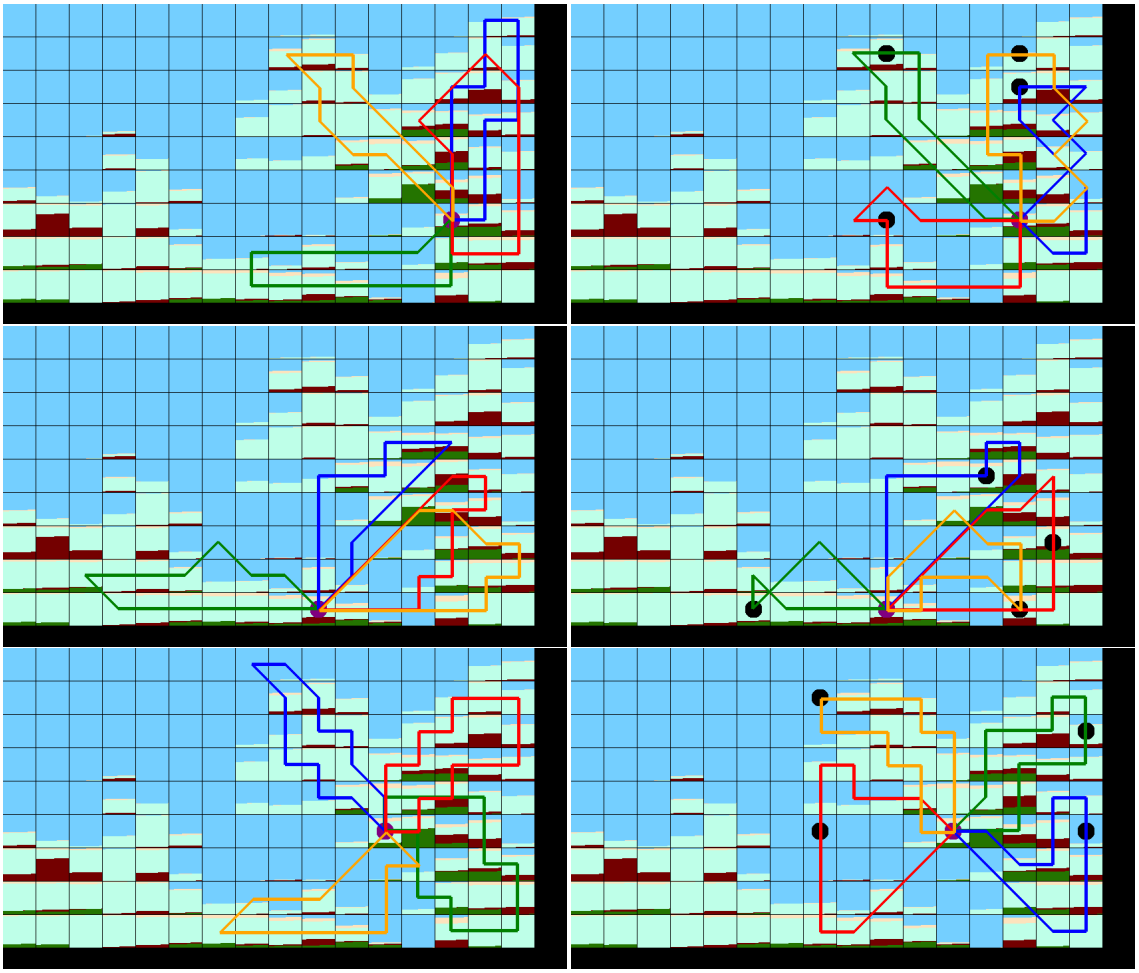


Figure A.10: Results on instance 10.

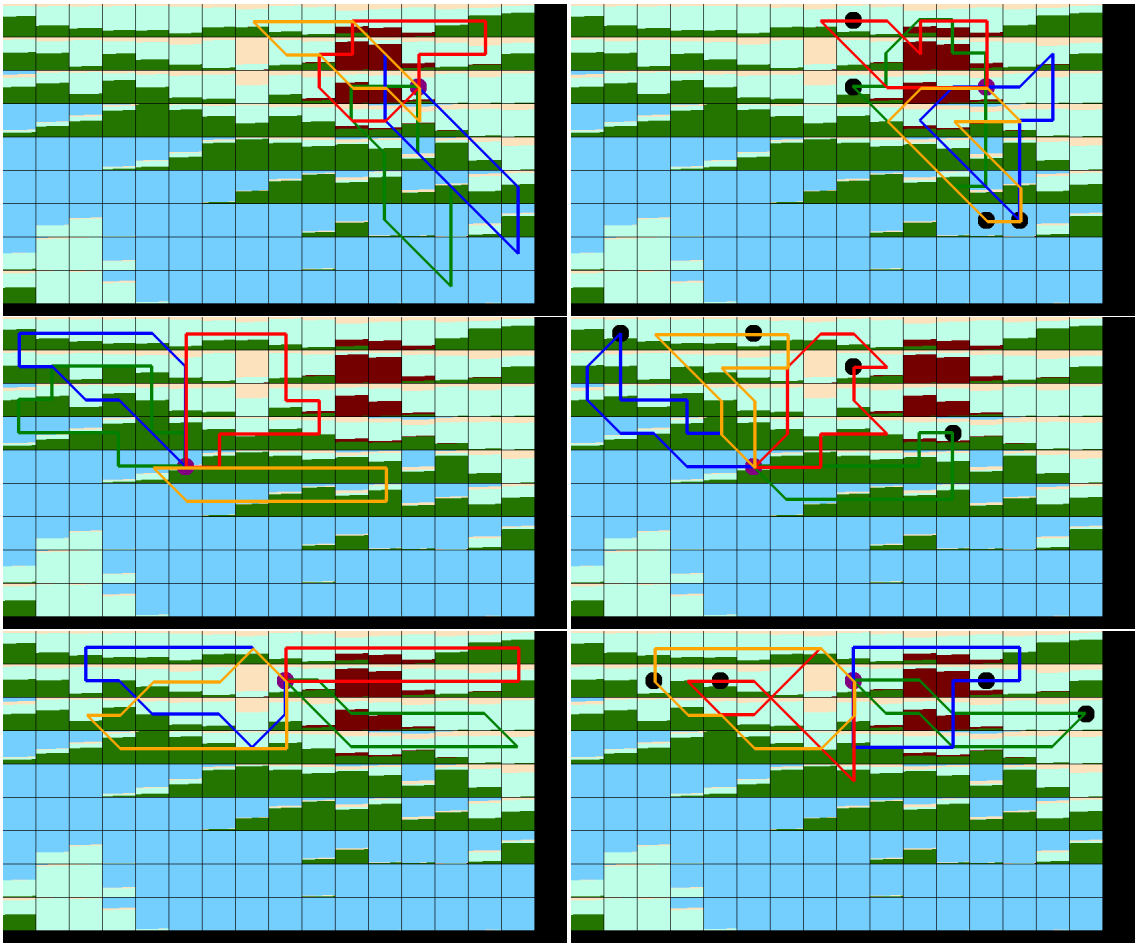


Figure A.11: Results on instance 11.

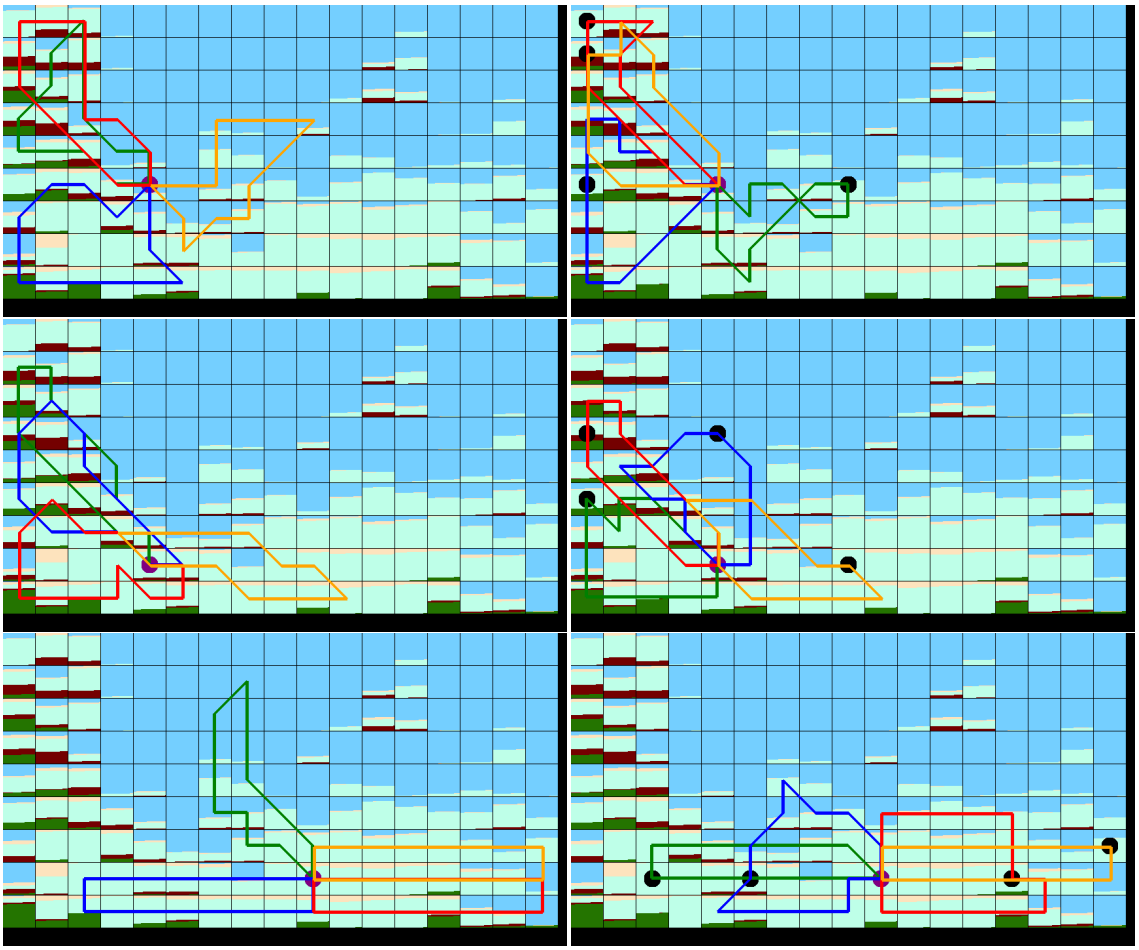


Figure A.12: Results on instance 12.

References

[1] I.-M. Chao, B. L. Golden, and E. A. Wasil, "The team orienteering problem," *European journal of operational research*, vol. 88, no. 3, pp. 464–474, 1996.

[2] P. Vansteenwegen, W. Souffriau, and D. V. Oudheusden, "The orienteering problem: A survey," *Eur. J. Oper. Res.*, vol. 209, pp. 1–10, 2011.

[3] P. C. Gilmore and R. E. Gomory, "A linear programming approach to the cutting-stock problem," *Operations Research*, vol. 9, pp. 849–859, 1961.

[4] G. Laporte, "The vehicle routing problem: An overview of exact and approximate algorithms," *European journal of operational research*, vol. 59, no. 3, pp. 345–358, 1992.

[5] D. Pisinger and S. Ropke, "A general heuristic for vehicle routing problems," *Computers & operations research*, vol. 34, no. 8, pp. 2403–2435, 2007.

[6] J. Kytöjoki, T. Nuortio, O. Bräysy, and M. Gendreau, "An efficient variable neighborhood search heuristic for very large scale vehicle routing problems," *Computers & operations research*, vol. 34, no. 9, pp. 2743–2757, 2007.

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

[8] A. Otto, N. Agatz, J. Campbell, B. Golden, and E. Pesch, "Optimization approaches for civil applications of unmanned aerial vehicles (uavs) or aerial drones: A survey," *Networks*, vol. 72, no. 4, pp. 411–458, 2018.

[9] P. Toth and D. Vigo, *Vehicle routing: problems, methods, and applications*. SIAM, 2014.

[10] R. Lahyani, M. Khemakhem, and F. Semet, "Rich vehicle routing problems: From a taxonomy to a definition," *European Journal of Operational Research*, vol. 241, no. 1, pp. 1–14, 2015.

[11] E. F. Flushing, M. Kudelski, L. M. Gambardella, and G. A. Di Caro, "Connectivity-aware planning of search and rescue missions," in *2013 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*. IEEE, 2013, pp. 1–8.

[12] Y. Marchukov and L. Montano, "Communication-aware planning for robot teams deployment," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 6875–6881, 2017.

[13] S. Hayat, E. Yanmaz, C. Bettstetter, and T. X. Brown, "Multi-objective drone path planning for search and rescue with quality-of-service requirements," *Autonomous Robots*, vol. 44, no. 7, pp. 1183–1198, 2020.

[14] E. I. Grøtli and T. A. Johansen, "Path planning for uavs under communication constraints using splat! and milp," *Journal of Intelligent & Robotic Systems*, vol. 65, no. 1, pp. 265–282, 2012.

[15] E. F. Flushing, L. M. Gambardella, and G. A. Di Caro, "Simultaneous task allocation, data routing, and transmission scheduling in mobile multi-robot teams," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 1861–1868.

[16] N. Mahdoui, V. Frémont, and E. Natalizio, "Cooperative frontier-based exploration strategy for multi-robot system," in *2018 13th Annual Conference on System of Systems Engineering (SoSE)*. IEEE, 2018, pp. 203–210.

[17] E. Yanmaz, "Dynamic relay selection and positioning for cooperative uav networks," *IEEE Networking Letters*, 2021.

[18] J. Scherer and B. Rinner, "Short and full horizon motion planning for persistent multi-uav surveillance with energy and communication constraints," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 230–235.

[19] T. Idota, E. Biagioni, and K. Binsted, "Swarm exploration of extraterrestrial lava tubes with ad-hoc communications," in *2018 6th IEEE International Conference on Wireless for Space and Extreme Environments (WiSEE)*. IEEE, 2018, pp. 163–168.

[20] M. Darvish, L. C. Coelho, and R. Jans, *Comparison of symmetry breaking and input ordering techniques for routing problems*. Faculté des sciences de l'administration, 2020.

[21] K. Hadj Salem and Y. Kieffer, "New symmetry-less ilp formulation for the classical one dimensional bin-packing problem," in *Computing and Combinatorics*, D. Kim, R. N. Uma, Z. Cai, and D. H. Lee, Eds. Cham: Springer International Publishing, 2020, pp. 423–434.

[22] G. Desaulniers, D. Pecin, and C. Contardo, "Selective pricing in branch-price-and-cut algorithms for vehicle routing," *EURO Journal on Transportation and Logistics*, vol. 8, no. 2, pp. 147–168, 2019, special Issue: Advances in vehicle routing and logistics optimization: exact methods. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2192437620300303

[23] M. Dror, "Note on the Complexity of the Shortest Path Models for Column Generation in VRPTW," *Operations Research*, vol. 42, no. 5, pp. 977–978, October 1994. [Online]. Available: https://ideas.repec.org/a/inm/oropre/v42y1994i5p977-978.html

[24] C. Tilk, A.-K. Rothenbächer, T. Gschwind, and S. Irnich, "Asymmetry matters: Dynamic half-way points in bidirectional labeling for solving shortest path problems with resource constraints faster," *European Journal of Operational Research*, vol. 261, no. 2, pp. 530–539, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0377221717302035

[25] G. Righini and M. Salani, "Symmetry helps: Bounded bi-directional dynamic programming for the elementary shortest path problem with resource constraints," *Discrete Optimization*, vol. 3, no. 3, pp. 255–273, 2006, graphs and Combinatorial Optimization. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1572528606000417

[26] S. K. Karakashian, B. Choueiry, and S. G. Hartke, "An algorithm for generating all connected subgraphs with k vertices of a graph," 2013.

[27] A. Luvisutto, M. Miolato, O. AlMaleki, A. Almarzooqi, E. Traversi, and G. De Masi, "Advanced modeling techniques for mission planning of marine multi-vehicles systems: What's next?" in *MTS/IEEE OCEANS - Hampton Roads*, 2022.