Multi-Drone Rescue Search in a Large Network*

Victor Gonzalez (victorgo@mit.edu)†[1] and Patrick Jaillet (jaillet@mit.edu)[2]

[1]Operations Research Center, MIT
[2]Operations Research Center, Electrical Engineering and Computer Science, MIT

February 2025

Abstract

Natural disasters are recurring emergencies that can result in numerous deaths and injuries. When a natural disaster occurs, rescue teams can be sent to help affected survivors, but deploying them efficiently is a challenge. Rescuers not knowing where affected survivors are located poses a significant challenge in delivering aid. With the development of new technologies, there are new possibilities to reduce this uncertainty, alleviating this challenge. One can first send out automated drones to locate affected survivors and then send rescue teams to their locations. We develop a model for the search process and construct mathematical methods to construct efficient search routes. We utilize a divide and conquer technique to determine the routes that are most likely to yield an efficient search. We combine this with our mathematical methods to construct efficient search routes in real-time and a method to update these routes in real-time as drones gather information.

Keywords: Nonlinear programming, large-scale optimization, routing, networks

1 Introduction

Natural disasters kill thousands of people each year globally. Deaths can be reduced by sending rescue teams to help those in vulnerable situations, but without proper information, this can be very inefficient. People could die because a rescue team was not sent to their location. If a rescue team is sent somewhere it is not needed, it will be unavailable to help someone else. The rescue process can be improved by first deploying automated search drones to find where rescue teams should be sent. The goal of this paper is to develop methods to construct and update efficient search routes in real-time to maximize the number of people found who need rescue services. In this paper, we consider a general natural disaster setting so that this work can be applied in a wide-range of instances where time is critical such as an earthquake or

*accepted, *European Journal of Operational Research*
†corresponding author

hurricane. Of high importance is the time-sensitive nature of this problem because time spent computing routes is time that could be spent finding someone who needs help from a rescue team.

We approach this problem by assuming that the area affected by the natural disaster can be discretized into "nodes" where someone may need rescue. These nodes can reference a building that was damaged in the disaster or a neighborhood where many people live or anything else. Discretizing the search area reduces the problem space, but it remains large because we must choose which node to send each drone to **and** how long to spend searching at each area **and** how these two change when drones gather new information. We reduce the problem further by defining the **static problem** which are the optimal search routes if we assume that drones cannot change their strategy upon gathering new information. Then, when drones gather new information, the operator can re-optimize the updated static problem. We develop a mixed-integer program (MIP) that can solved to compute a solution to the static problem, but with many nodes and drones this is still computationally challenging. We implement **simplifying policies** which are not guaranteed to be true at the optimal solution to the static problem but have favorable properties for a solution. These simplifying policies allow us to reformulate the MIP into one that can be solved quickly.

Since these MIPs must be solved multiple times, the overall computation time of each MIP must be considered. We exploit the fact that consecutive static problems are similar to each other by using route-limited optimization to limit the routes that drones can take to those most likely to yield good solutions. We do this by introducing a pre-processing step in which we use a divide and conquer heuristic to compute many solutions in parallel and use the routes that drones take most frequently to be those drones are allowed to take in route-limited optimization.

The remainder of the paper is organized as follows. In Section 2, we begin with a brief review of related work. In Section 3, we introduce the mathematical model that we use to represent this search problem. We then formulate a method that constructs optimal plans for drones in this model. In Section 4, in order to solve this model in real-time, we present simplifying methods which can be used to prune the feasible region of the method to solutions that are most likely to yield a good solution so approximate solutions can be computed quickly. We then present steps to update these solutions as the operator learns new information from the drones. In Section 5, we run computational experiments and discuss how our methods perform in a real-time optimization setting. The main contributions of this paper are to introduce a rigorous mathematical model for the problem under consideration, we construct a MIP that we can solve to construct good search efforts for drones in this model, and propose algorithmic methods that can improve computation to quasi real-time.

2 Literature review

Classical Search Problem: The first fundamental work on search problems was published by Brown [4]. The author finds an optimal search plan for the allocation of a searcher's effort to maximize the probability of finding a single target moving between nodes in discrete time intervals. The author solves this by exploiting the fact that at each of the discrete time intervals, the optimal search effort to allocate is the effort if you assume that the target was not found at a previous time step. This makes sense because if the target was found at a previous time step, it does not matter if it is found at the current time step. Our problem similarly has discrete nodes where people can be located. However, in our problem we have multiple targets and an objective based on maximizing the number of people found, rather than the probability of finding someone, and our search plan adapts upon receiving new information.

Search Problems with Multiple Targets: Simonin et al. [20] look at the problem of maximizing the probability that multiple targets are all found. The authors solve this problem by formulating the problem as a max-min problem to allocate search effort to maximize the probability that the target with minimum probability of discovery (given that search effort) is found. Lau et al. [13] address a similar question which incorporates the travel time of searchers to travel from one node to another. When a searcher arrives at a node, they can either search it or move to an adjacent node. The authors solve this problem using dynamic programming. Wong et al. [26] look at the problem of maximizing the sum of the cumulative detection probability of each target. They solve this by using Bayesian filtering to update the distribution for each target and a rolling time horiozon to solve the search control problem. Hou et al. [12] consider search scenarios where all targets must be found in the shortest time. They partition the search space into a grid where UAVs and targets each can occupy a node. UAVs can move between nodes, but their movement is constrained by the angle they are facing. They use a multi-agent reinforcement learning algorithm to solve this problem where each UAV is an agent. In our paper, we also consider search for multiple targets, but we formulate our objective differently in the way we formulate the objective based on the number of targets found.

Large-Scale Search Problems: Simonin et al. [21] consider the problem of searching for a moving target in problems that are large compared to the number of resources available. The authors approach this problem with a hierarchical, two-level approach. At the higher level, the region is split up into subsets of the search region, and at the lower level, sensors are allocated to these subsets. Delavarnhe et al. [6] look at a similar problem where the searcher has access to multiple mobile sensors that they use to maximize

the probability of finding the target. The authors solve this problem without dividing the search region into several subsets of the search region by constructing an algorithm which iteratively adds resources to nodes over several steps. Booth et al. [1] look at the problem of maximizing the cumulative probability of discovering a target when UAVs have limited charge. UAVs can be recharged at mobile recharging vehicles (MRVs). They define a set of routes that UAVs and MRVs are allowed to take and develop algorithms using a MIP approach and constraint programming approach to choose the best routes from the set. Similar to [21], we simplify the large problem by breaking it into small problems, but the exact method is different. Similar to [1] we look at a route-based approach and limit our optimization to those most likely to yield good solutions.

Techniques in Typical Search Problems: A lot of work has been done on search problems with small details distinguishing them from each other. However, there are many recurring techniques that are used to solve similar search problems that we summarize here. One approach is to model the search problem as a set of nodes that need to be searched an constructing a convex optimization problem that can be solved to find how much effort should be allocated to searching each node [7, 22]. Another is using evolutionary metaheuristics such as motion encoded swarm optimization and ant colony optimization [16, 17]. Alternatively, a Bayesian approach can be used to update the probability density of the target [2, 3, 8, 11]. A different option is using a Gaussian mixture model to identify good subsets of the continuous space to search [15, 28]. A new perspective is looking at a "coverage problem" where rather than directly searching for a target, UAVs take paths that "cover" the region [14, 15, 27, 31]

Strategic Planning during Natural Disasters: While the focus of this paper is searching for people during a natural disaster, the overall goal is to rescue as many people as possible during a natural disaster. We now shift our attention to papers focused on strategic planning during natural disasters. Hentenryck et al. [24] address the single commodity allocation problem in which a commodity can be stored at various nodes in a network to satisfy demand in the event of a disaster. They approach this problem by separating the problem into three stages. They start by determining which repositories should store the resource. Then, they compute the routes to be used for each repository to deliver the commodity to the customers. In the last stage, they determine the best way that the fleet of vehicles should be distributed to complete these routes as quickly as possible. Pillac et al. [18] look at the problem of determining evacuation routes from evacuated nodes to safe nodes in the event of a disaster. Edges have time limits for when they can be used as well as capacity. They approach this problem by applying a conflict-based path generation

algorithm. They compute paths to be added to the master problem by solving a multiple-origins, multiple destinations shortest path problem. Zheng and Ling [29] look at the problem of emergency transportation planning with the goal of delivering relief supplies to targets in a fast and efficient manner. This paper decomposes this plan into the steps of task allocation, resource allocation, delivery scheduling, and vehicle routing. They iteratively solve these subproblems together to reach an end result near pareto-optimal. Zheng et al. [30] look at an assignment problem in which rescue teams which can perform various jobs are assigned to different rescue tasks. These rescue teams must perform three jobs in order at each of the target positions, and each rescue team has different capabilities to accomplish each task. They approach the two problems of minimizing the total completion time and minimizing the total risk. They develop an algorithm to solve this problem using biogeography based optimization and adding in new migration and mutation operators. Reyes-Rubiano et al. [19] look at the problem of exploring a road network in the aftermath of a natural disaster for disrupted and functional roads. In the aftermath of a natural disaster, there can be road disruptions that prevent emergency services from serving victims. However, UAVs can be used to determine what victims can be served by emergency vehicles (if there is a route of undisturbed roads from the disaster management center to the victim). The authors develop online algorithms with the goal of minimizing the route length necessary to determine the accessibility of all victims.

Use of Drones in Logistics: We last turn our attention to research specifically in the use of drones, as these are the tool that we will use to search. Haidari et al. [10] use simulation to analyze the impact of using unmanned aerial vehicles to transport vaccines in underdeveloped regions. They compare the required costs to a traditional multi-tiered land transport system and showed that drones can reduce costs and improve vaccine availability. Ghelichi et al. [9] look at the problem of using drones to serve a set of demand points such that they minimize the total completion time. They use a reduction method to remove candidate routes that are dominated by others. In addition to humanitarian causes, drones can also be used in commercial logistics tasks such as package delivery, working in collaboration with traditional delivery methods [5, 23, 25].

3 Our model

While there could be survivors who do not need rescue, rescuers do not need to know where they are. For brevity, we use "survivors" to refer only to those in need of rescue.

Node assumptions: We model this problem as a network by partitioning the physical world into nodes where survivors might be in set N home-base nodes in set H. There is an an edge of length d_{ij} between any pair of nodes i and j. For each node $i \in N$, we define a random variable $S_i \in \{0, 1\}$ such that

$$S_i = \begin{cases} 1 & \text{if there is a survivor at node } i \\ 0 & \text{otherwise} \end{cases}$$

Drone assumptions: The operator has a fleet of m drones divided into K types and there are m_k drones of type k. All drones of type k have constant travel speed v_k and T_k battery life. We assume drones have access to constant, instantaneous communication with a centralized operator. Each drone ℓ of type k begins searching from a stationary home-base $h_{k\ell} \in H$ at time $t = 0$. They can return to any home-base to replace their battery, and replacement takes negligible time. There is no limit to how many times this can be done. At the end of the search time $t = T$, all drones must be at a home-base node.

Search assumptions: Finding survivors at one node could indicate survivors at nearby nodes, but a complete formulation of these relationships would require $2^{|N|}$ combinations. To facilitate the set-up and computations in practical settings, we assume S_i are independent. For each node $i \in N$, we define p_i to be the prior probability belief that a survivor is at node i, i.e., $\mathbb{P}(S_i = 1) = p_i$. We assume $0 < p_i < 1$ without loss of generality. In some settings, drones can immediately learn whether or not there are survivors at a node upon arrival e.g. if the drone is searching a building and survivors are outside or if the building is completely unaffected by the earthquake. For each node $i \in N$, we define $q_{i1}, q_{i0} \geq 0$ to be the probability that a drone immediately learns whether a survivor is at node i. This only applies to the first drone arriving at the node and does not depend on the drone type. In settings where this does not occur, we can define $q_{i1} = q_{i0} = 0$. In some settings, nodes cannot be searched for the entire search duration e.g. if a building is still being impacted by an earthquake or it is structurally damaged and will fall soon. For each node $i \in N$, we define t_i^{min} and t_i^{max} to be the times at which node i becomes available and unavailable respectively and we assume that these values are known. We define search rates $w_{ki}^1 \geq 0$ and $w_{ki}^0 \geq 0$ for searching when there is a survivor at node i or not, respectively, at node i. The higher the search rate, the faster the drone can learn S_i. We define the probability that a drone k learns whether there is a survivor at node i after searching for duration t:

$$\begin{cases} 1 - \exp\left(-w_{ki}^1 t\right) & \text{if } S_i = 1 \\ 1 - \exp\left(-w_{ki}^0 t\right) & \text{if } S_i = 0 \end{cases}$$

When $w^1_{ki} > 0$ or $w^0_{ki} > 0$, the average time to learn whether there is a survivor is $1/w^1_{ki}$ or $1/w^0_{ki}$ respectively. In settings where drones cannot confirm that there are no survivors at a node, we can define $w^0_{ki} = 0$. Intuitively, if multiple drones are searching the same node, their search efforts can be interpreted to "add up" together. We apply this intuition by assuming the probability of learning S_i is independent for each drone. The connection between the intuition and assumption can be verified in the following theorem:

Theorem 3.1. *Suppose that we have m_k drones with search rates w^1_{ki} and w^0_{ki} for all drone types k and nodes i. If each drone ℓ of type k searches node i for time $t_{k\ell i}$, the probability that at least one drone learns whether or not there is a survivor at node i is:*

$$\begin{cases} 1 - \exp\left(\sum_{k=1}^{K}\sum_{\ell=1}^{m_k} -w^1_{ki}t_{k\ell i}\right) & \text{if } S_i = 1 \\ 1 - \exp\left(\sum_{k=1}^{K}\sum_{\ell=1}^{m_k} -w^0_{ki}t_{k\ell i}\right) & \text{if } S_i = 0 \end{cases}$$

Further, if the true value of S_i is not learned by any drones, the posterior probability that $S_i = 1$ becomes:

$$\frac{p(1-q_{i1})\exp\left(\sum_{k=1}^{K}\sum_{\ell=1}^{m_k} -w^1_{ki}t_{k\ell i}\right)}{p(1-q_{i1})\exp\left(\sum_{k=1}^{K}\sum_{\ell=1}^{m_k} -w^1_{ki}t_{k\ell i}\right) + (1-p)(1-q_{i0})\exp\left(\sum_{k=1}^{K}\sum_{\ell=1}^{m_k} -w^0_{ki}t_{k\ell i}\right)}$$

Proof. If $S_i = 1$, the probability that S_i is not be learned by any drone is

$$\prod_{k=1}^{K}\prod_{\ell=1}^{m_k}\exp\left(-w^1_{ki}t_{k\ell i}\right) = \exp\left(\sum_{k=1}^{K}\sum_{\ell=1}^{m_k} -w^1_{ki}t_{k\ell i}\right)$$

This means that the probability that at least one drone learns S_i is

$$1 - \exp\left(\sum_{k=1}^{K}\sum_{\ell=1}^{m_k} -w^1_{ki}t_{k\ell i}\right)$$

If $S_i = 0$, the proof follows the same steps where w^1_{ki} is replaced with w^0_{ki}. Let X be the event where no drone learns about S_i. We can use Bayes' theorem to compute the new probability for S_i:

$$\begin{aligned} \mathbb{P}(S_i = 1|X) &= \frac{\mathbb{P}(X|S_i = 1)\cdot\mathbb{P}(S_i = 1)}{\mathbb{P}(X|S_i = 1)\cdot\mathbb{P}(S_i = 1) + \mathbb{P}(X|S_i = 0)\cdot\mathbb{P}(S_i = 0)} \\ &= \frac{p(1-q_{i1})\exp\left(\sum_{k=1}^{K}\sum_{\ell=1}^{m_k} -w^1_{ki}t_{k\ell i}\right)}{p(1-q_{i1})\exp\left(\sum_{k=1}^{K}\sum_{\ell=1}^{m_k} -w^1_{ki}t_{k\ell i}\right) + (1-p)(1-q_{i0})\exp\left(\sum_{k=1}^{K}\sum_{\ell=1}^{m_k} -w^0_{ki}t_{k\ell i}\right)} \end{aligned}$$

□

At the end of the search period, the operator must choose whether or not to send a rescue team to each node. If S_i is learned, this decision is trivial, but if not, the operator must make a decision based on the updated probability from Bayes' theorem. Some nodes might be "more important" to search if they represent an area that more people live or work. We define the utility that the operator receives for a true positive, false negative, true negative, and false positive detection at node i as U^i_{TP}, U^i_{FN}, U^i_{TN}, and U^i_{FP} respectively. Positive refers to the event that there are survivors and negative refers to the event that there are no survivors. The objective is to maximize the sum of the utilities of each node. We also assume that for every node i, $U^i_{TP} > U^i_{FN}$ and $U^i_{TN} > U^i_{FP}$ to match real-world incentives.

Fact 1. *Suppose that node i has utilities U^i_{TP}, U^i_{FN}, U^i_{FN}, and U^i_{FP}. Define*

$$\gamma_i = \frac{U^i_{TN} - U^i_{FP}}{(U^i_{TN} - U^i_{FP}) + (U^i_{TP} - U^i_{FN})}$$

If at time $t = T$, the operator has belief $p_i > \gamma_i$ that a survivor is at node i i.e. $S_i = 1$, then it is optimal to diagnose that there is a survivor at node i. If $p_i < \gamma_i$, then it is optimal to diagnose that there are no survivors at node i i.e. $S_i = 0$.

Proof. The operator achieves utility:

$$\begin{cases} p_i U^i_{TP} + (1 - p_i) U^i_{FP} & \text{if they diagnose that there is a survivor at node } i \\ p_i U^i_{FN} + (1 - p_i) U^i_{TN} & \text{if they diagnose that there is not a survivor at node } i \end{cases}$$

If $p_i > \gamma_i$,

$$\begin{aligned} p_i &> \gamma_i \\ p_i &> \frac{U^i_{TN} - U^i_{FP}}{(U^i_{TN} - U^i_{FP}) + (U^i_{TP} - U^i_{FN})} \\ p_i U^i_{TP} + (1 - p_i) U^i_{FP} &> p_i U^i_{FN} + (1 - p_i) U^i_{TN} \end{aligned}$$

which means that it is optimal to diagnose that there is a survivor at node i. If $p_i < \gamma_i$, the proof proceeds as before where " $>$ " is replaced with " $<$ ". □

Consequently, we can assume without loss of generality that $U^i_{TN} = U^i_{FN} = 0$ for all nodes $i \in N$ because the "threshold probability" γ_i can be expressed in terms of only $(U^i_{TN} - U^i_{FP})$ and $(U^i_{TP} - U^i_{FN})$. As a final modeling consideration, all computations should be achievable in real-time. We aim at developing methods that can effectively address networks of sizes up to 96 nodes and 48 drones. As a reference, refer to Table 15 in the supplementary materials for terms that define the model.

4 Optimization methods

4.1 Methodology

An optimal solution consists of an initial route (any sequence of nodes that the drone visits) for each drone, the time allocated to searching each node, **and** how these change when a drone detects whether or not a survivor is at a node. To guarantee optimality, these must be determined before any searching occurs. Since we consider continuous time, there are infinitely many times at which a drone can detect whether or not a survivor is at a node leading to an intractable problem that cannot be solved in real-time.

We simplify this by considering **static methods**, methods that compute routes and time allocations as if they could not change them as drones learn new information. We refer to this problem of maximizing the expected utility given this assumption as the **static problem**. This simplification drastically reduces the decision space to one that is tractable. In real-life, drones are not limited to the routes and time allocations that they are assigned at the beginning. When the operator learns whether or not a survivor is at a node, routes and time allocations can be updated by solving an updated static problem.

We first develop a MIP that is equivalent to the static problem. However, even with few nodes and drones, this MIP can still be computationally challenging to solve. To facilitate a solution to the static problem, we implement **simplifying policies** which are not guaranteed to be true at the optimal solution to the static problem but have favorable properties for a solution. These simplifying policies allow us transform the MIP into one that can be solved quicker.

Since these MIPs must be solved multiple times, the overall computation time of each MIP must be considered. We exploit the fact that consecutive static problems are similar to each other by using route-limited optimization to limit the routes that drones can take to those most likely to yield good solutions. We do this by introducing a pre-processing step in which we use a divide and conquer heuristic to compute many solutions in parallel and use the routes that drones take most frequently to be those drones are allowed to take in route-limited optimization.

4.2 General method

We first construct two MIPs that can be solved to compute an optimal solution to the static problem. We refer to a round as any time that the drone takes a route which leaves a node in H and returns to a node in H. We define a trivial route to be any route (i, i) for $i \in H$, and we define a trivial round to be any round where the drone takes a trivial route. Since replacing the battery requires negligible time, we can assume that each drone has maximum charge at the beginning of each round.

We use Table 1 to define terms based on the model parameters:[1]

Table 1: Terms for defining the general method

$\mathcal{P}$	All routes that begin and end at a node in H
L	Length of the longest route in $\mathcal{P}$
$\mathcal{P}_{is}$	Routes $P \in \mathcal{P}$ such that i is the s^{th} node in P
$\mathcal{P}_i$	Routes $P \in \mathcal{P}$ such that $i \in P$
$\mathcal{P}_i^1$ $(\mathcal{P}_i^{-1})$	Routes $P \in \mathcal{P}$ such that i is the first (last) node
$\mathcal{P}_{ij}^s$	Routes $P \in \mathcal{P}$ such that $P_s = i$ and $P_{s+1} = j$
$\mathcal{P}_{ij}^{-1}$	Routes $P \in \mathcal{P}$ such that $P_{\|P\|-1} = i$ and $P_{\|P\|} = j$

Define the binary decision variables:

$$x_{k\ell Pr} = \begin{cases} 1 & \text{if drone } \ell \text{ of type } k \text{ takes route } P \text{ in its } r^{th} \text{ round} \\ 0 & \text{otherwise} \end{cases}$$

and auxiliary decision variables to express the utility function:

$$z_i = \begin{cases} 1 & \text{if node } i \text{ is visited by at least one drone during any round} \\ 0 & \text{otherwise} \end{cases}$$

We use Table 2 to define decision variables related to search effort time:

Table 2: Search effort time decision variables for the general method

$t_{k\ell i}$	Time that drone ℓ of type k spends searching node i
$t_{k\ell ris}$	Time that drone ℓ of type k spends searching node $i \in N$ in round r if it is the s^{th} node visited in the route
$t_{k\ell ris}^{min}$ $(t_{k\ell ris}^{max})$	Time at which drone ℓ of type k begins (ends) searching node $i \in N$ in round r if it is the s^{th} node visited in the route
$t_{k\ell r0}^{min}$ $(t_{k\ell r0}^{max})$	Time at which the ℓ^{th} drone of type k arrives at (leaves) the home-base node before round r

Let y_{i1} and y_{i0} be the auxiliary decision variables representing the expected utility at node i if the operator diagnoses there is a survivor or not when drones do not confirm the presence or absence of

[1]There are some routes in the defined set $\mathcal{P}$ that can never be present in the optimal solution to the static problem. We removed these notes in the computational experiments. However, the process for finding these routes is beyond the scope of this paper because of the simplifying policies in the next section. Further details can be found in the supplementary materials

survivors. Since we assume that $U^i_{TN} = U^i_{FN} = 0$ for $i \in N$, we can express them as:

$$y_{i0} = p_i z_i \left(1 - (1 - q_{i1}) e^{-\sum_{k=1}^{m} \sum_{\ell=1}^{m_k} w^1_{ki} t_{k\ell i}}\right) U^i_{TP}$$

$$y_{i1} = p_i U^i_{TP} + (1 - p_i) \left(z_i (1 - q_{i0}) e^{-\sum_{k=1}^{m} \sum_{\ell=1}^{m_k} w^0_{ki} t_{k\ell i}} + (1 - z_i)\right) U^i_{FP}$$

Last define the maximum number of rounds that drones can take to be $R = \max_{k \in [K], i \in H, j \in N} \left\lfloor \frac{T \times s_k}{2 d_{ij}} \right\rfloor$. Next, we define constraint sets which are used in multiple MIPs. To limit redundancy, define an index set $\mathcal{I} = [K] \times [m_k] \times [R]$. Define the set of feasible routes that drones can take:

$$\mathcal{X}_1 = \left\{ \mathbf{x} \;\middle|\; \begin{aligned} x_{k\ell P1} = 0 \;\forall k \in [K],\; \ell \in [m_k],\; \forall P \in \mathcal{P} \text{ if } P_1 \neq h_{k\ell} \\ \sum_{P \in \mathcal{P}^1_{h_{k\ell}}} x_{k\ell P1} = 1 \forall k \in [K],\; \forall \ell \in [m_k] \\ \sum_{P \in \mathcal{P}^1_h} x_{k\ell P(r+1)} = \sum_{P \in \mathcal{P}^{-1}_h} x_{k\ell Pr} \;\forall h \in H,\; \forall k \in [K],\; \forall \ell \in [m_k],\; \forall r \in [R-1] \end{aligned} \right\}$$

These constraints ensure that each drone begins its first round at its initial home-base node (1), only one route is taken in the first round (2), and each drone takes exactly one route in other rounds beginning at the node that the previous round ended at (3). Define the set of feasible time allocations:

$$\mathcal{X}_2 = \left\{ \mathbf{t} \;\middle|\; \begin{aligned} t_{k\ell ris} \leq t^{max}_{k\ell ris} - t^{min}_{k\ell ris} \;\forall (k,\ell,r) \in \mathcal{I},\; \forall i \in N,\; \forall s \in [L] \setminus \{1, L\} \\ t^{max}_{k\ell ris} \leq t^{max}_i \;\forall (k,\ell,r) \in \mathcal{I},\; \forall i \in N,\; \forall s \in [L] \setminus \{1, L\} \\ t^{min}_{k\ell ris} \geq t^{min}_i \;\forall (k,\ell,r) \in \mathcal{I},\; \forall i \in N,\; \forall s \in [L] \setminus \{1, L\} \end{aligned} \right\}$$

These constraints ensure the time allocated to searching a node equals the time between when it arrives and leaves (1) and a drone can only search a node while it is available (2-3). Define the set of combinations of routes and time allocations where a drone can only search a node after it arrives from the previous node:

$$\mathcal{X}_3 = \left\{ (\mathbf{x}, \mathbf{t}) \;\middle|\; \begin{aligned} t^{min}_{k\ell ri2} - t^{max}_{k\ell r0} \geq d_{ji}/v_k - M(1 - x_{k\ell Pr}) \;\forall (k,\ell,r) \in \mathcal{I},\; \forall (i,j) \in N \times H,\; \forall P \in \mathcal{P}^1_{ji} \\ t^{min}_{k\ell ri(s+1)} - t^{max}_{k\ell rjs} \geq d_{ji}/v_k - M(1 - x_{k\ell Pr}) \;\forall (k,\ell,r) \in \mathcal{I},\; \forall i \in N,\; \forall j \in N, \\ \forall s \in \{2, \ldots, |P| - 2\},\; \forall P \in \mathcal{P}^s_{ji} \\ t^{min}_{k\ell(r+1)0} - t^{max}_{k\ell ri(|P|-1)} \geq d_{ji}/v_k - M(1 - x_{k\ell Pr}) \;\forall (k,\ell,r) \in \mathcal{I},\; \forall (i,j) \in H \times N,\; \forall P \in \mathcal{P}^{-1}_{ji} \end{aligned} \right\}$$

Example 1. *To assist in visualizing $\mathcal{X}_1$, $\mathcal{X}_2$, and $\mathcal{X}_3$ consider drone 1 of type 1 takes the following plan with $R = 4$ (the drone takes the trivial route (H_2, H_2) in rounds 3 and 4):*

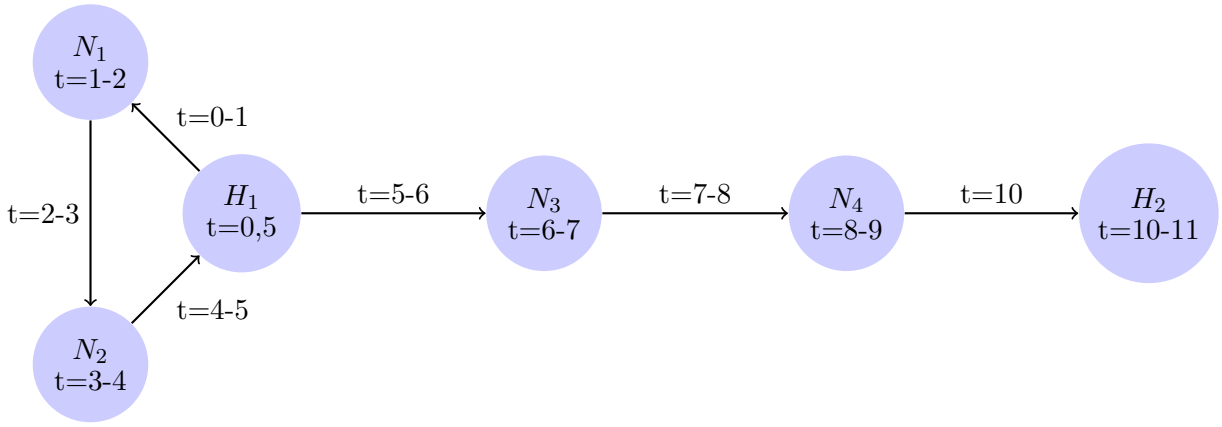

Figure 1: Illustration for Example 1

$\mathcal{X}_1$ enforces that the drone starts from H_1 and starts its next round where the last round finished (H_1, then H_2, then H_2). $\mathcal{X}_2$ bounds t_{11ris} by 1 for all appropriate r, i, s and enforces nodes can only be searched when they are available (not represented in the example). $\mathcal{X}_3$ enforces that the drone cannot search a node until it travels from the previous node e.g. it takes a unit of time to travel from N_1 to N_2.

The solution to the following nonlinear mixed-integer optimization problem solves the static problem:

$$
\begin{aligned}
\max \quad & \textstyle\sum_{i\in N} \max\{y_{i0}, y_{i1}\} && && (1)\\
& y_{i0} = p_i z_i \left(1-(1-q_{i1})e^{-\sum_{k=1}^{m}\sum_{\ell=1}^{m_k} w^1_{ki} t_{k\ell i}}\right) U^i_{TP} && && (1a)\\
& y_{i1} = p_i U^i_{TP} + (1-p_i)\left(z_i(1-q_{i0})e^{-\sum_{k=1}^{m}\sum_{\ell=1}^{m_k} w^0_{ki} t_{k\ell i}} + (1-z_i)\right) U^i_{FP} && && (1b)\\
& z_i \le \textstyle\sum_{k=1}^{K}\sum_{\ell=1}^{m_k}\sum_{P\in\mathcal{P}_i}\sum_{r=1}^{R} x_{k\ell Pr} && \forall i \in N && (2)\\
& t_{k\ell ris} \le T_k \textstyle\sum_{P\in\mathcal{P}_{is}} x_{k\ell Pr} && \forall (k,\ell,r)\in\mathcal{I},\ \forall i\in N,\ \forall s\in[L]\setminus\{1,L\} && (3)\\
& t_{k\ell i} = \textstyle\sum_{r=1}^{R}\sum_{s=1}^{L} t_{k\ell ris} && \forall k\in[K],\ \forall \ell\in[m_k],\ \forall i\in N && (4)\\
& t^{min}_{k\ell(r+1)0} - t^{max}_{k\ell r0} \le T_k && \forall (k,\ell,r)\in\mathcal{I} && (5)\\
& \mathbf{x}\in\mathcal{X}_1,\ \mathbf{t}\in\mathcal{X}_2,\ (\mathbf{x},\mathbf{t})\in\mathcal{X}_3 && && (6)\\
& 0 \le t_{k\ell i}, t^{min}_{k\ell ris}, t^{max}_{k\ell ris}, t^{min}_{k\ell r0}, t^{max}_{k\ell r0} \le T && \forall (k,\ell,r)\in\mathcal{I},\ \forall i\in N,\ \forall s\in[L]\setminus\{1,L\} &&\\
& x_{k\ell Pr}\in\{0,1\} && \forall (k,\ell,r)\in\mathcal{I},\ \forall P\in\mathcal{P} &&\\
& z_i\in\{0,1\} && \forall i\in N &&
\end{aligned}
$$

Here, (1) represents the total utility which is the sum of the utilities of individual nodes. (1a) and (1b) define the objective variables. (2) ensures that z_i can only be 1 if there is a drone that takes a route which includes that node. (3) ensures that a drone can only allocate time to search node $i \in N$ as the s^{th} node in its route in round r if i is the s^{th} node in the route it takes in round r. (4) defines the total time traveled by drone k at node i. (5) ensures that the total time spent in a route is not more than the battery life of a drone. (6) ensures that the routes and time allocations are feasible as defined before.

The objective is non-linear and not guaranteed to be concave because it is the sum of maximums of two terms. This can be challenging for finding a global optimal solution with a solver. We can instead modify the formulation by moving the nonlinear terms into the constraints as follows. Let y_i be an auxiliary variable representing the unconditional expected utility at node i. Let λ_i be an auxiliary variable such that $\lambda_i = 1$ if the operator diagnoses that there is a survivor, i.e., if

$$p_i > \frac{U^i_{TN} - U^i_{FP}}{(U^i_{TN} - U^i_{FP}) + (U^i_{TP} - U^i_{FN})}$$

and $\lambda_i = 0$ otherwise (see Fact 1). The mixed-integer optimization problem can be reformulated with nonlinear terms moved to the constraints by replacing objective (1) with

$$\max \sum_{i \in N} y_i$$

where y_i is an auxiliary decision variable for the objective of node i and adding constraints $(1c)$

$$y_i \leq y_{i1}\lambda_i + y_{i0}(1 - \lambda_i) \; \forall i \in N$$

that ensure $y_i = \max\{y_{i0}, y_{i1}\}$ for all nodes. The objective is linear, but the feasible region is no longer guaranteed to be convex, meaning that it can be difficult to determine whether a local optimal is the global optimal. We find the general method with linear objective to be preferable because while both have non-linear constraints, this method has a linear objective.

4.3 Simplifying policies

The general method yields optimal solutions to the static problem, but the MIP can be difficult to compute when there are many nodes and drones. We propose three simplifying policies that reduce the feasible region to improve the computation time. These policies may reduce the objective of the computed solution, but due to the time-sensitivity of the problem, the improvement in computational time outweighs this.

4.3.1 Drones can only search a node once per round

Some settings have an optimal route where a drone visits the same node multiple times as seen in the supplementary materials, but this is uncommon. This motivates our first simplifying policy:

Policy 1: A drone can only visit each node in N once per round.

This policy can be implemented by making the following changes to the MIP in the general method:

1. Replace $\mathcal{P}$ with $\mathcal{P}'$ containing only routes where nodes in N appears at most once
2. Replace $\mathcal{X}_2$ and $\mathcal{X}_3$ with $\mathcal{X}_2'$ and $\mathcal{X}_3'$ by replacing $t^{min}_{k\ell ris}$, $t^{max}_{k\ell ris}$, and $t_{k\ell ris}$ with $t^{min}_{k\ell ri}$, $t^{max}_{k\ell ri}$, and $t_{k\ell ri}$
3. Replace $t^{min}_{k\ell ris}$, $t^{max}_{k\ell ris}$, and $t_{k\ell ris}$ with $t^{min}_{k\ell ri}$, $t^{max}_{k\ell ri}$, and $t_{k\ell ri}$ in constraints (3) and (4)
4. Remove summation over s in constraint (4)

4.3.2 Limit on the number of nodes a drone can visit in a round and number of rounds

Intuitively, increasing the number of nodes a drone visits during a round decreases the time it can allocate to searching them. This intuition is supported by computational experiments on instances with few nodes and drones in which drones only visit a few nodes in each round. Additionally, the number of non-trivial rounds taken by drones will frequently be lower than the theoretical maximum. These motivate our second simplifying policy:

Policy 2: Each drone can visit at most N^{max} nodes in N in a single round and take at most R' rounds. We implement this by replacing $\mathcal{P}'$ with $\mathcal{P}''$ containing all routes in $\mathcal{P}'$ with at most N^{max} nodes in N and replacing R with R'. In our experiments, we choose $N^{max} = 2$ by varying N^{max} and comparing the marginal change in the objective and computational time. However, a higher N^{max} may be appropriate if computation time is less urgent or if the ratio of nodes to "drone-rounds" (mR') is higher (in which case the impact on the objective may be larger). If N^{max} is not chosen appropriately, this could lead to a model that does not yield good solutions or is too large to compute solutions quickly.

4.3.3 Only send a rescue team if the drone confirms survivors

In real-world settings, there is a limited supply of rescue teams that can be sent out for survivors so sending a rescue team to a node where there are no survivors can have devastating consequences. Intuitively, we expect that most nodes would have sufficiently low p_i such that the operator would not send a rescue team to a node if a drone was not sent there. Additionally, we would expect that the search rate for confirming a survivor at a node to be higher than the search rate for confirming that there is not a survivor at a node meaning that if no survivors are found at a node p_i can only decrease (see Theorem 3.1). Combining this intuition motivates our third simplifying policy:

Policy 3: A rescue team is only sent to nodes for which drones confirm there is a survivor

To implement this by replacing U^i_{FP} with $U^{i'}_{FP} = -\infty$ for all nodes $i \in N$. Under this condition, $y_{i1} = -\infty$

because we assume that $0 < p_i < 1$. This means

$$\max \sum_{i \in N} \max\{y_{i0}, y_{i1}\} = \max \sum_{i \in N} y_{i0}$$

This eliminates the need to introduce the y_{i1} non-linear terms and the constraints (1c) and variable λ_i in the general method with linear objective. We can further simplify the formulation by recognizing that

$$y_{i0} = \begin{cases} 0 & \text{if } z_i = 0 \\ p_i \left(1 - (1 - q_{i1}) e^{-\sum_{k=1}^{m} \sum_{\ell=1}^{m_k} w_{ki}^1 t_{k\ell i}}\right) U_{TP}^i & \text{otherwise} \end{cases}$$

Now, define the set of objective constraints:

$$\mathcal{X}_4 = \left\{ (\mathbf{x}, \mathbf{y}, \mathbf{t}) \;\middle|\; \begin{array}{r} y_i \le U_{TP}^i \sum_{k=1}^{K} \sum_{\ell=1}^{m_k} \sum_{P \in \mathcal{P}_i''} \sum_{r=1}^{R} x_{k\ell Pr} \;\forall i \in N \\ y_i \le p_i \left(1 - (1 - q_{i1}) e^{-\sum_{k=1}^{m} \sum_{\ell=1}^{m_k} w_{ki}^1 t_{k\ell i}}\right) U_{TP}^i \;\forall i \in N \end{array} \right\}$$

These constraints enforce that if a node is not visited by any drone, i.e. $z_i = 0$, the objective for that node must be 0, and the objective must always be bounded by

$$p_i \left(1 - (1 - q_{i1}) e^{-\sum_{k=1}^{m} \sum_{\ell=1}^{m_k} w_{ki}^1 t_{k\ell i}}\right) U_{TP}^i.$$

This value is strictly positive and less than U_{TP}^i so if $z_i = 0$, the first constraint is binding, and if $z_i = 1$, the second constraint is binding.

4.4 Simplified method

By combining the simplifying policies, we arrive at the simplified MIP:

$$\begin{array}{lll}
\max & \sum_{i \in N} y_i & (1) \\
\text{s.t.} & t_{k\ell ri} \le T_k \sum_{P \in \mathcal{P}_i''} x_{k\ell Pr} \quad \forall (k, \ell, r) \in \mathcal{I},\ \forall i \in N & (2) \\
& t_{k\ell i} = \sum_{r=1}^{R'} t_{k\ell ri} \quad \forall k \in [K],\ \forall \ell \in [m_k],\ \forall i \in N & (3) \\
& t_{k\ell(r+1)0}^{min} - t_{k\ell r0}^{max} \le T_k \quad \forall (k, \ell, r) \in \mathcal{I} & (4) \\
& \mathbf{x} \in \mathcal{X}_1,\ \mathbf{t} \in \mathcal{X}_2',\ (\mathbf{x}, \mathbf{t}) \in \mathcal{X}_3',\ (\mathbf{x}, \mathbf{y}, \mathbf{t}) \in \mathcal{X}_4 & (5) \\
& 0 \le t_{k\ell i}, t_{k\ell ri}^{min}, t_{k\ell ri}^{max}, t_{k\ell r0}^{min}, t_{k\ell r0}^{max} \le T \quad \forall (k, \ell, r) \in \mathcal{I},\ \forall i \in N & \\
& x_{k\ell Pr} \in \{0, 1\} \quad \forall (k, \ell, r) \in \mathcal{I},\ \forall P \in \mathcal{P}'' &
\end{array}$$

The simplified MIP trades off potential feasible solutions to construct a MIP that can be solved faster.

4.5 Scaling to larger problems

In this section, we introduce methods to solve larger problems with a target problem size up to 96 nodes and 48 drones. We begin by introducing the randomized divide and conquer method that splits the problem into smaller problems, each containing a subset of the nodes and drones that can be solved independently. This is a heuristic method that quickly constructs routes and time allocations for drones. While the obtained solutions are not the best, using many of them can give an idea of which routes are "high-value". We use the results of the randomized divide and conquer method as a pre-processing step to the route-limited optimization method which limits the routes that drones are allowed to take to only high-value routes.

4.5.1 Randomized divide and conquer method

The idea of the randomized divide and conquer method is to randomly construct groups of nodes and drones and then solve each group quickly using a heuristic. We start by randomly assigning nodes and drones to groups. For each group, we assign a route to each drone one drone at a time and one round at a time in a random order. A major advantage of this method is that computing routes and time allocations for drones in different groups are independent from each other so all groups can be solved simultaneously. In the following algorithm, we refer to the **single route optimization problem** where all routes are fixed to pre-defined routes or must be chosen from a set of trivial routes except for the route of one drone in one round. It can be constructed by replacing the decision variables in the simplified MIP corresponding to the fixed routes with their fixed value and removing the routes and time allocations for drones in rounds that are constrained to fixed routes. Let $g \geq 1$ to be the desired number of groups:

Algorithm 1: Randomized divide and conquer

Data: All network data; $g \geq 1$
Result: Feasible solution

Randomly partition N into g groups such that the j^{th} has $\left\lfloor \frac{|N|+j-1}{g} \right\rfloor$ nodes;
Randomly partition the drones into g such that the j^{th} has $|D_j| = \left\lfloor \frac{m+j-1}{g} \right\rfloor$ drones;
for *each group* i *in* $1, \ldots, g$ **do**
    Randomly order the drones in the group;
    $fixed\ routes \leftarrow \varnothing$;
    for *each drone in the group* **do**
        for *each round in* $1, \ldots, R'$ **do**
            Solve the single route optimization problem for the drone and round using $fixed\ routes$;
            Add the route for the drone and round to $fixed\ routes$;
    $(x^*, t^*)_i \leftarrow$ the solution to the last single route optimization problem;
return $(x^*, t^*)_i$ for $i = 1, \ldots, g$;

4.5.2 Route-limited optimization

The route-limited optimization algorithm uses the solutions generated by the randomized divide and conquer method to choose high value routes. The algorithm first constructs heuristic solutions using Algorithm 1. We first choose the heuristic solutions that give the best objective because they are the most indicative of high value routes. Then, we take the routes that occur most frequently in these solutions. As with Algorithm 1, each randomized heuristic is independent so they can be computed in parallel. Let $g \geq 1$ be the number of groups used in Algorithm 1. Let *sims* be the number of heuristic solutions to be constructed using Algorithm 1. Let X be the number of these solutions used and let Y be the number of routes considered for each drone type in the final problem in each round.

Algorithm 2: Route selection

Data: All network data; $g \geq 1$; sims; X; Y
Result: Set of routes $(\mathcal{P}^{out}_{11}, \ldots, \mathcal{P}^{out}_{KR})$ that drones of type k can take in round R

for *each sim in* $1, \ldots, sims$ **do**
  $x^{sim}, obj^{sim} \leftarrow$ routes used in random feasible solution and the associated objective;
$S \leftarrow$ the X solutions with the highest objective;
for *drone type* k *in round* r **do**
  $\mathcal{P}^{out}_{kr} \leftarrow$ the Y routes that are taken most frequently by drones of type k in round r in the solutions in S and all trivial routes;
return $(\mathcal{P}^{out}_{11}, \ldots, \mathcal{P}^{out}_{KR})$;

The simplified MIP can be updated to construct the route-limited MIP by replacing $\mathcal{P}''_i$ with $\mathcal{P}^{out}_{kr}$, replacing $\mathcal{P}^1_h$ with $\mathcal{P}^1_{krh}$ (routes in $\mathcal{P}^{out}_{kr}$ starting at h), and replacing $\mathcal{P}^{-1}_h$ with $\mathcal{P}^{-1}_{krh}$ (routes in $\mathcal{P}^{out}_{kr}$ ending at h) in constraints (2) and (5) of the simplified MIP.

4.6 Re-optimization

As drones search nodes, they improve their estimates of whether or not there is a survivor at each node, and they can use this to improve their routes and/or time allocations at each node. The updates to routes and time allocations can be solving a modified version of route-limited optimization MIP. This section describes how the model parameters can be updated. The routes chosen in Algorithm 2 all start from the a node in H, but drones might not be at a home-base node during re-optimization. The section continues to show how these routes can be updated without repeating pre-processing.

Updating probabilities, nodes, battery, current node, and round: If a node is visited by a drone for the first time, and the drone does not immediately detect whether or not there is a survivor, the

probability that there is someone in need of rescue at that node is updated using Bayes' theorem:

$$p_i = \frac{(1-q_{i1})p_i}{(1-q_{i1})p_i + (1-q_{i0})(1-p_i)}$$

and the probability that a drone learns immediately upon arrival whether or not there is someone in need of rescue is updated to 0 i.e. $q_{i1} = q_{i0} = 0$. For every other node i that had been searched since the last re-optimization, we update the probability using Bayes' theorem. Let t_{ki} be the total time that drones of type k have searched node i. We update

$$p_i = \frac{\exp\left(-\sum_{k=1}^{m} w_{ki}^1 t_{ki}\right) p_i}{\exp\left(-\sum_{k=1}^{m} w_{ki}^1 t_{ki}\right) p_i + \exp\left(-\sum_{k=1}^{m} w_{ki}^0 t_{ki}\right)(1-p_i)}$$

as we see in Theorem 3.1. If a drone confirms whether or not survivors are at a node or not, the node is removed from the network. The battery for each drone is set to the total charge minus the time elapsed since the last recharge. The current node of each drone is updated based on the node it is at. Last, we track how many times each drone returned to a home-base node to know which round it is in. This is important for knowing which routes should be considered as time-windows imply that make some nodes are best searched in certain rounds.

While new information is constantly updated as drones search nodes, we must choose which changes are substantial enough to warrant re-optimization. There is a tradeoff in how often we re-optimize with better solutions if we re-optimize more often and faster computation if we re-optimize less often. Two such instances are when a node is visited for the first time, and when we learn whether there is someone in need of rescue at a node:

1. When a node is visited for the first time, we simplify re-optimization by only re-optimizing over the time that drones spend at each node without changing the routes that drones take.
2. In the instance where we learn whether or not there is someone in need of rescue, we expedite optimization by only allowing drones to change their routes if they are currently visiting or are scheduled to visit the node for which this information was learned.

In each case, by not doing a full re-optimization, we can benefit from new information while trying to minimize redundant computations. Finally, if drones are between nodes during re-optimization, we enforce that drones must arrive at their destination node before changing their route to prevent situations where drones are constantly changing which node they are going to without ever arriving at one. This assumption also cleans up route selection so there is no need to create artificial nodes to start routes.

4.6.1 Updating $\mathcal{P}_{kr}^{out}$

Algorithm 2 can be used to choose routes for the route-limited optimization method. Each of these routes begins and ends at a home-base node, but this can be problematic when re-optimizing if a drone is not at a home-base node. Additionally, there may be nodes present at the beginning but are no longer necessary for drones to search either because it was learned whether or not there is a survivor or time t_i^{max} has passed. In this section, we discuss how we can update $\mathcal{P}_{kr}^{out}$ as we learn new information.

In the optimal solution, a drone would never take a route containing only nodes that are no longer available. We update Algorithm 2 by updating line 8 to say that P must include at least one node in N which has not been removed. Additionally, for each node i such that there is a drone at the node or a drone traveling to that node, we include a truncated copy of each route containing that node. This truncated route begins with i and includes all nodes that are still available. We use truncated routes so that the objective of the re-optimization problem can be at least as good as that of the previous optimization problem, avoiding a situation where re-optimizing makes the solution worse. Let $N' \subseteq N$ be the set of nodes that are still available to search.

Algorithm 3: Update route selection

Data: All network data; $g \geq 1$; sims; X; Y; *locs*: nodes where a drone is located
Result: Set of routes $(\mathcal{P}_{11}^{out}, \ldots, \mathcal{P}_{KR}^{out})$ used for optimization

1 **for** *each sim in* $1, \ldots, sim$ **do**
2 $x^{sim}, obj^{sim} \leftarrow$ routes used in random feasible solution and the associated objective;
3 $S \leftarrow$ the X solutions with the highest objective;
4 **for** *each* k *in* $1, \ldots, K$ *and* r *in* $1, \ldots, R$ **do**
5 $\mathcal{P}_{kr} \leftarrow$ the Y routes that are taken most frequently by drones of type k in round r that contain a node in N';
6 **for** *each* k *in* $1, \ldots, K$ *and* r *in* $1, \ldots, R$ **do**
7 $\mathcal{P}_{kr}^{out} \leftarrow \varnothing$;
8 **for** $P \in \mathcal{P}_{kr}$ **do**
9 Add a truncated copy of P;
10 **for** i *in* $2, \ldots, |P| - 1$ **do**
11 **if** $P_i \in locs$ **then**
12 Add a truncated copy of P starting from node i;
13 **return** $(\mathcal{P}_{11}^{out}, \ldots, \mathcal{P}_{KR}^{out})$;

5 Computational experiments

Throughout this section, we run computational experiments on randomized network instances of various sizes to test the change in computation time and objective. We use network size to refer to the number of

nodes and drones in the network. We use small networks (<10 nodes) to test the slower methods that trade off computation time to improve the objective. We use small networks to test the trade off in the objective when we use methods that decrease the computation time. We use large networks (≥48 nodes) to test how computation time and objective change as we increase the number of nodes i.e. do these algorithms "scale". The networks in these experiments use two home-base nodes and maintain a ratio of two nodes for each drone. For each network size, we randomly construct 200 network instances where each network instance has randomized node and drone parameters. For networks of each size, we construct 200 random instances and take the average objective. We assume that all nodes, including the home-base nodes are distributed uniformly at random in a 144km × 144km square. Once, generated, we can construct d_{ij} based one the geometric distance. The remaining parameters can be found in Table 3. These parameters are based on parameters discussed with our collaborators, and the exponential parameter for w^1_{ki} and w^0_{ki} is tuned so that the objectives generated by our algorithms would be about half of the maximum objective if the operator knew precisely whether or not there was someone in need of rescue at each node. For all networks, we assume that we have half as many drones as there are nodes. Experiments were run using the MIT Engaging cluster using 8 GB of memory per experiment.

Table 3: Parameters for computational experiments

n, m	Vary by problem size
H	{1,2}
p_i	Drawn independently from $Unif(.4, .6)$
q_{i1}, q_{i0}	Drawn independently from $Unif(.1, .2)$
S_i	Drawn independently from $Bernoulli(p_i)$
T	4 hours
t_i^{min}	Drawn independently from $Unif$(0, 2 hours)
t_i^{max}	Drawn independently from $Unif$(2 hours, 4 hours)
K	2
m_k	$\frac{m}{2}$
v_k	40 m/s
T_k	2 hours
$h_{k\ell}$	1 for the first $\frac{m_k}{2}$ of each drone type and 2 for the remaining $\frac{m_k}{2}$
w^1_{ki}	Drawn independently from $Exp(3)$ (search rate is in hours)
w^0_{ki}	Drawn independently from $Exp(3)$ (search rate is in hours)
U^i_{TP}	Drawn independently from $Unif(70, 130)$
U^i_{FP}	(For testing general method) -10^6

5.1 Evaluation of simplifying policies

In this section, we evaluate the simplifying policies given in Section 4.3. We test variations where we include or do not include:

1. Simplifications to the feasible region from limiting the length of routes and imposing that a node can only be visited by a drone once in a round.

2. Simplification to the objective function.

The four methods we test are summarized in Table 4. In General and Objective, we define the set of routes $\mathcal{P}$ to be all routes beginning at a node in H and ending at a node in H which do not satisfy the condition in Lemma B.1 in the supplementary material and could be completed by a drone within the time allocated by its battery life.

Table 4: Simplifying experiments

	Feasible region simplifications	No feasible region simplifications
Objective function simplifications	Simplified	Objective
No objective simplifications	Route length limited (RLL)	General

To test the simplifying policies, we must choose the best R' and N^{max} such that each method can be solved consistently. We say a method can be solved consistently for networks of a certain size if the method can be solved for 95% of our randomized instances of that size with MIP gap $< 1\%$ within 10 minutes. We say that we have the best parameters if increasing R' or N^{max} does not improve the objective. We find that for networks with four or fewer nodes, for each method, the best parameters that we can use to solve methods consistently have $N^{max} = 2$ and $R' = 2$. We cannot solve the problem consistently on networks with six or more nodes. These parameters make sense intuitively because the total search time is twice as much as the max charge of drones and there are twice as many nodes as drones. These parameters are used in all the computational experiments because they perform best on small networks, and the methods used for solving larger networks are built using solutions to smaller networks i.e. divide and conquer. Table 5 records the average objective Table 6 records the average and standard deviation of computation time.

Table 5: Average objectives of computed optimal solution

Nodes	General	RLL	Objective	Simplified
2	32.21	32.18	32.24	32.21
4	n/a	82.80	n/a	82.80
6	n/a	n/a	n/a	n/a

Table 6: Average time (st. dev) in seconds to compute optimal solution

Nodes	General	RLL	Objective	Simplified
2	7.99 (9.53)	7.11 (9.52)	2.30 (1.98)	2.75 (2.16)
4	n/a	187.95 (156.81)	n/a	182.89 (127.83)
6	n/a	n/a	n/a	n/a

All four methods give objective within 1% of each other with two nodes, and with four nodes RLL and Simplified give the same results. We interpret this to mean the simplifying policies do not cut out the best feasible solutions. Adding in the route length limiting and objective simplifications allow for significant increase in computation power. We conclude that the simplified method is the best of our simplification methods due to its fast speed and comparable objective.

5.2 Evaluation of randomized divide and conquer methods

In this section, we test Algorithm 1 on the same small networks used for Tables 5 and 6 to see how it compares in small networks. The random nature of this method leads to variation in the computation time and objective so we run Algorithm 1 200 times for each of the 200 network instance. Table 7 records the average computation time, average objective, and maximum objective for a total of 40,000 runs of Algorithm 1. Computation time refers to the average time across all 40,000 runs to compute the route and time allocation for the divide and conquer group that takes longest to compute. Since each group in Algorithm 1 is independent, this represents the average computation time to run Algorithm 1 if the routes and time allocation for each each group is computed in parallel. The average objective refers to the average objective across all 40,000 runs of the randomized divide and conquer method. The maximum objective refers to the average across all 200 network instances of the maximum objective across the 200 times the randomized divide and conquer method was run on this instance. This represents the average objective if we run the divide and conquer method 200 times and take the best solution.

Table 7: Randomized divide and conquer method on small networks

Nodes	Groups	Average computation time (s)	St. dev computation time (s)	Average objective	Maximum objective
2	1	.29	.16	29.31	29.31
4	1	.92	.29	74.88	76.65
4	2	.55	.17	61.94	75.62
6	1	1.95	.39	113.83	117.37

The maximum objective attained from Algorithm 1 is lower than that given by the general and simplified

methods even with two nodes and one group which may seem counter-intuitive. This results from Algorithm 1 calculating the route a drone takes in each round independently from what route is taken in future rounds. The simplified method, on the other hand, computes the route taken and time allocation in the same MIP. With two and four nodes, we see that Algorithm 1 greatly improves the computation time at a small loss to the objective, and it can be solved on network instances of a size that the simplified method cannot be solved on. We conclude that Algorithm 1 serves its purpose as a means of choose high value routes.

5.2.1 Comparing similar randomized divide and conquer methods

In this section, we evaluate variations on Algorithm 1 when we alter:

1. Whether the route that a drone takes in each round is computed one round at a time (sequential) or if all rounds are computed in the same MIP (concurrent).

2. Whether multiple drones can visit the same node in a single round (unrestricted) or not (restricted).

By combining these two elements, we create four randomized divide and conquer methods. Algorithm 1 refers to the sequential-unrestricted method. We test these four methods on large networks with 48 nodes. This is large enough that the results should be representative of networks with 96 nodes, but is small enough that we can test the methods that take longer. In each of these methods, we divide the network into 6 groups. Since there are only 8 nodes in each group, this is still in a similar size to the results that we observe in 5.1. As in the previous section, we compute 200 randomized feasible solutions using these four methods and compare the results. Table 8 records the computation time and objective for the various divide and conquer methods.

Table 8: Comparing variations of Algorithm 1

Method	Average computation time (s)	St. dev computation time (s)	Average objective	Max objective
sequential-unrestricted	23	2	1006	1070
sequential-restricted	14	2	668	785
concurrent-unrestricted	2542	142	1032	1095
concurrent-restricted	730	71	945	1009

The concurrent-restricted method is strictly dominated by the sequential-unrestricted method which has faster computation time and higher average and max objective. The sequential-restricted method has faster computation time for much lower average and max objective compared to the sequential-unrestricted

method. The concurrent-unrestricted method has higher average and max objective for much higher computation time compared to the sequential-unrestricted method method. We conclude that the sequential-unrestricted method, i.e. Algorithm 1 gives the best balance between computation time and objective.

5.2.2 Computational impact of modifying the number of groups

In this section, we test Algorithm 1 as we vary the number of groups used to determine the number of groups that should be used by Algorithm 1 when pre-processing for the route-limited optimization method. We compare the average time and objective for each number of groups, g. Algorithm 1 is only run on each network interest once as with fewer groups this computation can take a long time. This method is tested on network instances with 48 nodes. Table 9 records the computation time and objective as the number of groups is varied.

Table 9: Computational impact of modifying the number of groups

Number of groups (g)	Nodes per group	Average computation time (s)	Average objective
6	8	20.51	924.48
4	12	34.53	967.01
3	16	93.47	993.10
2	24	544.96	1023.34
1	48	6161.72	1058.78

There is a trade-off between the computation time and average objective. As the number of groups increases, the size of each group decreases and vice versa. As the size of groups increases, the effect of randomness in the random partitioning has a bigger impact on the solution leading to a lower average objective. We conclude that the best number of nodes per group is 12 because it gives the best balance between computation time and solution quality. We use 12 nodes per group when testing the route-limited optimization method in the following section. We keep as a constant the number of nodes per group instead of the number of groups as this leads to the most consistent computation time. With a different set of input parameters, a similar process can be repeated by testing the average computation time and objective and choosing the number of nodes per group at which the marginal increase in the computation time outweighs the marginal increase in the utility. Generally, with fewer drones, it is preferable to have more nodes in each group and vice versa.

5.3 Evaluation of route-limited optimization method

In this section, we test the route-limited optimization method on network instances with 48 nodes and 96 nodes. We use $N^{max} = 2$ and $R' = 2$ as they were optimal for Algorithm 1. In each experiment, we compute 200 randomized divide and conquer solutions which we use to choose the best routes in route-limited optimization method. We begin testing the route-limited optimization method on network instances with 48 nodes. First, we test the objective of solutions as we modify the number of solutions (X) and the number of routes (Y). In these experiments, the network instances are too large to compute the optimal solution to the route-limited optimization method. We limit the route-limited optimization method to two minutes of computation time to fairly compare the objective that can be achieved using any combination of these parameters and to get a better understanding of the computational power of this method in a real-world setting. There are 12 drones of each type so at most $12X$ routes can be used for each drone type.

Table 10: Route-limited optimization method objective as # of solutions and # of routes vary (48 nodes)

X \ Y	60	72	84	96	108	120	132	144	156
5	1024.59								
6	1026.76	1029.14							
7	1021.47	1025.34	1024.78						
8	1026.02	1025.51	1022.13	1023.83					
9	1031.76	1025.40	1020.41	1021.50	1022.23				
10	1029.80	1026.91	1019.30	1018.91	1019.11	1016.21			
11	1034.72	1028.80	1020.84	1016.12	1015.05	1016.15	1015.42		
12	1030.86	1024.88	1017.58	1010.33	1009.88	1009.28	1009.52	1009.61	
13	1033.55	1026.21	1015.99	1011.72	1010.09	1009.72	1009.53	1009.55	1009.63

By combining solutions of the randomized divide and conquer method, the route-limited optimization method can improve upon even the best solution obtained from Algorithm 1. Using the 11 best solutions and choosing to use 60 routes for each drone type in each round yields the best results, and these are the inputs that we use when testing re-optimization. The parameters that work best are dependent on the network parameters and the computational time so these will not be the best parameters in general.

In real-world settings, as discussed in Section 4.6, the operator can update the routes and time allocations of drones as they learn new information. The next experiments test the results of a simulated search process of networks. For each of the 200 network instances, we run 200 randomized simulations. In each simulation, we randomly assign whether or not a survivor is at each node and how long a drone has to

search a node to detect whether or not there is a survivor. We run two re-optimization experiments. We first run Algorithm 1 200 times to construct the set of candidate routes $\mathcal{P}_{kr}^{out}$ to be used for both experiments. The initial choice of routes and time allocation is determined by the route-limited optimization method using the 11 best solutions and choosing to use 60 routes for each drone type in each round. In the first experiment, which we call Fixed routes, drones can adjust the time allocated to searching nodes when new information is found. However, they must remain on the route assigned by route-limited optimization at the beginning. In the second experiment, which we call Non-fixed routes, drones are allowed to change their route using route-limited optimization when a drone arrives at a node for the first time or when a drone learns that there is a survivor at a node. If a drone arrives at a node for the first time, we update our solution in the same way as Fixed routes since this occurs often and the nodes in the network are the same. If a drone learns whether or not a survivor is at a node, we allow all drones to update their time allocations at each node and allow drones that were searching that node or planned to search that node to update their routes using route-limited optimization. However, **the 200 pre-processing solutions are not re-calculated**. Instead the candidate routes are updated by using Algorithm 3. If the previous route that a drone was taking is no longer a candidate route after the algorithm is run, the route is added to the set of candidate routes to ensure that re-optimization always leads to a solution that is at least as good. Table 11 compares the average objective of the two re-optimization approaches and compares them to the objective when no re-optimization is performed.

Table 11: Route-limited optimization with updates (48 nodes)

Static route-limited optimization	Fixed routes	Non-fixed routes
1031.66	1046.39	1060.82

Updating the routes and time-allocations of drones, the operator can improve the objective by ≈ 30 which translates to $\approx 30\%$ of a new node where drones learn that there is a survivor because the average U_{TP} for a node in these experiments is 100. This can represent an important benefit that can be gained by re-optimizing when new information is learned. Seeing that re-optimization does not have a bigger change in terms of the objective could imply that a method that optimizes the objective in the static problem is a smart way to approach the non-static, real-world version of this problem. We additionally repeat these experiment on networks with 96 nodes in Tables 12 and 13.

Table 12: Route-limited optimization method objective as # of solutions and # of routes vary (96 nodes)

X \ Y	12	24	36	48	60	72
1	1581.31	2005.75				
2	1521.90	1818.18	1978.75	1991.76		
3	1584.91	1802.35	1921.63	1975.33	1947.22	1891.12

Table 13: Route-limited optimization with updates (96 nodes)

Baseline	Fixed routes	Non-fixed routes
2004.02	2016.56	2037.59

Using just the routes used in the best solution, we obtain the best solution using route-limited optimization. This could be due to the size of the problem making the impact of adding in more candidate routes increase the size of the feasible region more than when running the route-limited optimization method on networks with 48 nodes. As we would expect, the operator can improve the objective by re-optimizing corresponding to $\approx 33\%$ of a new node where drones learn that there is someone in need of rescue. Last, we see that the objective found using the route-limited optimization method on networks with 96 nodes is roughly double that of the objective when running this method on networks with 48 nodes. This can represent the ability of this method to scale to larger problems which is what we expect.

5.4 Sensitivity to the number of drones

In this section, we test the route-limited optimization method when we vary the number of drones. To test this, we take a network with 48 nodes and alter the number of drones. We repeat the same methodology used throughout this section to compute the number of groups, the number of solutions used (X), and the number of routes used (Y). Table 14 indicates the change in the objective as we vary the number of drones.

Table 14: Route-limited optimization with updates varying number of drones (48 nodes)

Drones	Baseline	Fixed routes	Non-fixed routes
12	740.90	737.68	746.59
24	1031.66	1046.39	1060.82
48	1321.55	1350.65	1373.30

Table 14 shows that as the number of drones is increased, the objective increases as well, but there are

diminishing returns. This makes sense intuitively because as more drones are used to search, they will be searching nodes that are already being searched by other drones.

6 Conclusion

In this paper, we introduce several optimization methods to route drones to find survivors in the event of a natural disaster. We start by discretizing the areas that drones can search. Under this discretized model, we introduce two general methods that maximize the expected number of survivors. This is not sufficient because we need to be able to make decisions in real-time. We introduce simplifying policies preventing drones from visiting nodes multiple times in a round, visiting too many nodes in a round, and diagnosing a survivor if it cannot be confirmed. These policies restrict the feasible region to expedite the rate at which a solution is found with the simplified method. This allows us to quickly find solutions, but not at the level of real-time which is necessary when we look at re-optimization. We introduce the randomized divide and conquer method which takes advantage of parallel processing to partition the network into smaller networks on which it is faster to solve the simplified method. We use the best solutions obtained by the randomized divide and conquer method and use them to select candidate routes that are most likely to yield good solutions in the route-limited optimization method. After using this as a pre-processing at the beginning of the search, this allows us to perform re-optimizations in real-time. In our computational experiments, we conclude that the route-limited optimization method with non-fixed routes in re-optimization works best. Natural disasters can have a devastating impact on human lives', and our work should help assist in efficiently providing relief to those most in need.

One potential new research direction is using reinforcement learning techniques to improve choosing routes and time allocations. Our setting is too time-sensitive to implement this, but this may be beneficial in another setting. Another direction would be using a continuous 3-D model of the world, rather than representing it using a network. Further research could combine the search stage of finding survivors with the rescue stage of sending rescue teams to their locations or add in new nodes during the search process.

Acknowledgement: This work was partially supported by a DSTA-MIT grant DST000ECI20300823

References

[1] Booth, K. E., Piacentini, C., Bernardini, S., & Beck, J. C. (2020). Target search on road networks with range-constrained UAVs and ground-based mobile recharging vehicles. *IEEE Robotics and Automation*

Letters, 5(4), 6702-6709.

[2] Bourgault, F., Furukawa, T., & Durrant-Whyte, H. F. (2003, October). Coordinated decentralized search for a lost target in a Bayesian world. In *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)(Cat. No. 03CH37453)* (Vol. 1, pp. 48-53). IEEE.

[3] Bourgault, F., Furukawa, T., & Durrant-Whyte, H. F. (2006). Optimal search for a lost target in a bayesian world. *Field and Service Robotics: Recent Advances in Research and Applications*, 209-222.

[4] Brown, S. S. (1980). Optimal search for a moving target in discrete time and space. *Operations Research, 28*(6), 1275-1289.

[5] Das, D. N., Sewani, R., Wang, J., & Tiwari, M. K. (2020). Synchronized truck and drone routing in package delivery logistics. *IEEE Transactions on Intelligent Transportation Systems, 22*(9), 5772-5782.

[6] Delavernhe, F., Jaillet, P., Rossi, A., & Sevaux, M. (2021). Planning a multi-sensors search for a moving target considering traveling costs. *European Journal of Operational Research, 292*(2), 469-482.

[7] Eagle, J. N., & Yee, J. R. (1990). An optimal branch-and-bound procedure for the constrained path, moving target search problem. *Operations Research, 38*(1), 110-114.

[8] Furukawa, T., Bourgault, F., Lavis, B., & Durrant-Whyte, H. F. (2006, May). Recursive Bayesian search-and-tracking using coordinated UAVs for lost targets. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.* (pp. 2521-2526). IEEE.

[9] Ghelichi, Z., Gentili, M., & Mirchandani, P. B. (2021). Logistics for a fleet of drones for medical item delivery: A case study for Louisville, KY. *Computers & Operations Research, 135*, 105443.

[10] Haidari, L. A., Brown, S. T., Ferguson, M., Bancroft, E., Spiker, M., Wilcox, A., ... & Lee, B. Y. (2016). The economic and operational value of using drones to transport vaccines. *Vaccine, 34*(34), 4062-4067.

[11] Hollinger, G., Singh, S., Djugash, J., & Kehagias, A. (2009). Efficient multi-robot search for a moving target. *The International Journal of Robotics Research, 28*(2), 201-219.

[12] Hou, Y., Zhao, J., Zhang, R., Cheng, X., & Yang, L. (2023). UAV swarm cooperative target search: A multi-agent reinforcement learning approach. *IEEE Transactions on Intelligent Vehicles.*

[13] Lau, H., Huang, S., & Dissanayake, G. (2005, August). Optimal search for multiple targets in a built environment. In *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 3740-3745). IEEE.

[14] Lin, L., & Goodrich, M. A. (2009, October). UAV intelligent path planning for wilderness search and rescue. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 709-714). IEEE.

[15] Lin, L., & Goodrich, M. A. (2014). Hierarchical heuristic search using a Gaussian mixture model for UAV coverage planning. *IEEE Transactions on Cybernetics, 44*(12), 2532-2544.

[16] Morin, M., Abi-Zeid, I., & Quimper, C. G. (2023). Ant colony optimization for path planning in search and rescue operations. *European Journal of Operational Research, 305*(1), 53-63.

[17] Phung, M. D., & Ha, Q. P. (2020). Motion-encoded particle swarm optimization for moving target search using UAVs. *Applied Soft Computing, 97*, 106705.

[18] Pillac, V., Van Hentenryck, P., & Even, C. (2016). A conflict-based path-generation heuristic for evacuation planning. *Transportation Research Part B: Methodological, 83,* 136-150.

[19] Reyes-Rubiano, L., Voegl, J., Rest, K. D., Faulin, J., & Hirsch, P. (2021). Exploration of a disrupted road network after a disaster with an online routing algorithm. *OR Spectrum, 43*(1), 289-326.

[20] Simonin, C., Le Cadre, J. P., & Dambreville, F. (2008, June). A common framework for multitarget search and cross-cueing optimization. In *2008 11th International Conference on Information Fusion* (pp. 1-8). IEEE.

[21] Simonin, C., Le Cadre, J. P., & Dambreville, F. (2009). A hierarchical approach for planning a multisensor multizone search for a moving target. *Computers & Operations Research, 36*(7), 2179-2192.

[22] Stewart, T. J. (1979). Search for a moving target when searcher motion is restricted. *Computers & Operations Research, 6*(3), 129-140.

[23] Troudi, A., Addouche, S. A., Dellagi, S., & Mhamedi, A. E. (2018). Sizing of the drone delivery fleet considering energy autonomy. *Sustainability, 10*(9), 3344.

[24] Van Hentenryck, P., Bent, R., & Coffrin, C. (2010, June). Strategic planning for disaster recovery with stochastic last mile distribution. In *International Conference on Integration of Artificial Intelligence (AI) and Operations Research (OR) Techniques in Constraint Programming* (pp. 318-333). Berlin, Heidelberg: Springer Berlin Heidelberg.

[25] Wang, Z., & Sheu, J. B. (2019). Vehicle routing problem with drones. *Transportation Research Part B: Methodological, 122,* 350-364.

[26] Wong, E. M., Bourgault, F., & Furukawa, T. (2005, April). Multi-vehicle Bayesian search for multiple lost targets. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation* (pp. 3169-3174). IEEE.

[27] Xu, A., Viriyasuthee, C., & Rekleitis, I. (2011, May). Optimal complete terrain coverage using an unmanned aerial vehicle. In *2011 IEEE International Conference on Robotics and Automation* (pp. 2513-2519). IEEE.

[28] Yao, P., Zhu, Q., & Zhao, R. (2020). Gaussian mixture model and self-organizing map neural-network-based coverage for target search in curve-shape area. *IEEE Transactions on Cybernetics, 52*(5), 3971-3983.

[29] Zheng, Y. J., & Ling, H. F. (2013). Emergency transportation planning in disaster relief supply chain management: a cooperative fuzzy optimization approach. *Soft Computing, 17,* 1301-1314.

[30] Zheng, Y. J., Ling, H. F., & Xue, J. Y. (2014). Disaster rescue task scheduling: An evolutionary multiobjective optimization approach. *IEEE Transactions on Emerging Topics in Computing, 6*(2), 288-300.

[31] Zhu, K., Han, B., & Zhang, T. (2021). Multi-UAV distributed collaborative coverage for target search using heuristic strategy. *Guidance, Navigation and Control, 1*(01), 2150002.

A Model terminology

Table 15: Terms to define the model

N	Nodes where survivors might be
H	Home-base nodes
d_{ij}	Distance between i and j
S_i	Whether or not a survivor is at node i
p_i	Prior belief that a survivor is at node i
q_{i1} (q_{i0})	Probability that a drone immediately learns there is (not) a survivor at node i
T	Time at which drones must be at a home-base
t_i^{min} (t_i^{max})	Time before (after) which node i cannot be searched
m	Number of drones
K	Number of drone types
m_k	Number of drones of type k
v_k	Travel speed of drone k
T_k	Battery life of drone k
$h_{k\ell}$	Starting home-base of drone ℓ of type k
w_{ki}^1 (w_{ki}^0)	Search rate of drones of type k at node i if there is (not) a survivor
U_{TP}^i, U_{FN}^i, U_{TN}^i, U_{FP}^i	Utility for true positive, false negative, true negative, false positive

B Pruning the routes in $\mathcal{P}$

When considering $\mathcal{P}$, some routes may never be included in an optimal solution. In this section, we develop a sufficient condition for a route to be absent in an optimal solution, and so to be safely removed from $\mathcal{P}$.

Lemma B.1. *Suppose that we have a route* $H_1, \ldots, x, \ldots, y, \ldots, x, \ldots, y, \ldots, H_2$ *where*

- $H_1, H_2 \in H$
- H_1 *may or may not be the same node as* H_2
- $x, y \in N$
- *Each "..." refers to a, potentially empty, set of nodes between the two nodes on either side*
- *We define* z_1, z_2, z_3, z_4, z_5 *to be these five sets (in the same order that they appear)*

Then, a sufficient condition for a route to be a sub-optimal solution is

$$(z_2 \subseteq z_3 \cup z_4 \cup z_5 \wedge z_4 \subseteq z_1 \cup z_2 \cup z_3) \vee$$
$$(z_2 \cup z_3 \subseteq z_4 \cup z_5) \vee (z_3 \cup z_4 \subseteq z_1 \cup z_2)$$

Proof. Define $t_{x1}, t_{y1}, t_{x2}, t_{y2}$ to be the time the drone spends at x and y the first and second time. Suppose that a route satisfies this condition. We can split this into three cases

Case 1: $z_2 \subseteq z_3 \cup z_4 \cup z_5 \wedge z_4 \subseteq z_1 \cup z_2 \cup z_3$. This implies that every node visited in z_2 could be visited later, as long as they are searched before the time the drone reaches y the first time. Similarly, every node in z_4 could be visited earlier as long as they are searched before the time the drone reaches x the second time. If $t_{y1} > t_{x2}$, we can construct the route $H_1, z_1, x, z_2, y, z_3, z_4, y, z_5, H_2$ where z_i represents the drone visiting each node in between the two nodes in the original route. Let z_i^1 and z_i^{-1} be the first and last nodes in z_i respectively. For simplicity, let s with no subscript refer to the speed of the drone. Here the drone spends $t_{x1} + t_{x2}$ at x, $t_{y1} - t_{x2}$ at y the first time, $t_{x_2} + t_{y_2} + \varepsilon_1$ at y the second time, and the same time at all other nodes where

$$0 < \varepsilon_1 < \frac{d_{z_3^{-1}x} + d_{xz_4^1} - d_{z_3^{-1}z_4^1}}{s}$$

Note here that drone leaves y the first time at the same time in both of these routes. If $t_{y1} < t_{x2}$, we can construct the route $H_1, z_1, x, z_2, z_3, x, z_4, y, z_5, H_2$. Here the drone spends $t_{x1} + t_{y1} + \varepsilon_2$ at x the first time, $t_{x2} - t_{y1}$ at x the second time, and $t_{y1} + t_{y2}$ at y, and the same time at all other nodes where

$$0 < \varepsilon_2 < \frac{d_{z_2^{-1}y} + d_{yz_3^1} - d_{z_2^{-1}z_3^1}}{s}$$

If $t_{y1} = t_{x2}$, we can construct the route $H_1, z_1, x, z_2, z_3, z_4, y, z_5, H_2$. Here the drone spends $t_{x1} + t_{x2} + \varepsilon_2$ at x, $t_{y1} + t_{y2} + \varepsilon_1$ at y, and the same time at all other nodes.

Case 2: $z_2 \cup z_3 \subseteq z_4 \cup z_5$. We can construct the route $H_1, z_1, x, z_2, y, z_3, z_4, y, z_5, H_2$ where the drone spend $t_{x1} + t_{x2} + \varepsilon_2$ at node x and the same time at all other nodes.

Case 3: $z_3 \cup z_4 \subseteq z_1 \cup z_2$. We can construct the route $H_1, z_1, x, z_2, z_3, x, z_4, y, z_5, H_2$ where the drone spends $t_{y1} + t_{y2} + \varepsilon_1$ at y, and the same time at all other nodes. □

C Optimal solution in which a drone visits the same node multiple times

Example 2. *As illustrated in Figure 2, consider a network with* $N = \{1, 2\}$ *and* $H = \{0\}$ *with* $d_{01} = d_{12} = 1$ *and* $d_{02} = 2$. *Let* $t_1^{min} = 1$, $t_1^{max} = 6$, $t_2^{min} = 3$, $t_2^{max} = 4$ *and let* $p_i = .5$, $q_{i1} = q_{i0} = 0$, $U_{TP}^i = 1$, $U_{FN}^i = U_{TN}^i = 0$, *and* $U_{FP}^i = -\infty$ *for* $i \in N$. *Further let there be one drone with 7 units of time for battery and 1 unit of speed. Let* $w_{11}^0 = w_{12}^0 = 0$, $w_{11}^1 << w_{12}^1$. *For* w_{12}^1 *sufficiently larger than* w_{11}^1, *the optimal strategy includes the drone spending time at node 2. Suppose that the drone searches node 2 between times* t_1 *and* t_2 *such that* $3 \leq t_1 \leq t_2 \leq 4$. *For any of these strategies, we can maximize the time that the drone searches node 1 by having the drone search in times* $[1, t_1 - 1] \cup [t_2 + 1, 6]$.

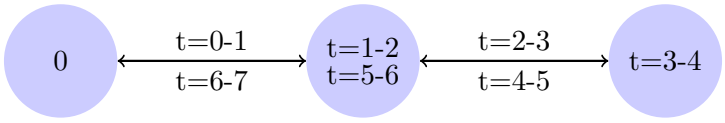

Figure 2: Illustration for Example 2