Chase C. Murray and Mark H. Karwan, “A Branch-and-Bound-Based Solution Approach for Dynamic Rerouting of Airborne Platforms,” *Naval Research Logistics*, 60 (2), 141-159, 2013.

http://dx.doi.org/10.1002/nav.21526

A Branch-and-Bound-Based Solution Approach for Dynamic Rerouting of Airborne Platforms

Abstract: This paper is a sequel to a recent paper that appeared in this journal, "An extensible modeling framework for dynamic reassignment and rerouting in cooperative airborne operations" [17], in which an integer programming formulation to the problem of rescheduling in-flight assets due to changes in battlespace conditions was presented. The purpose of this paper is to present an improved branch-and-bound procedure to solve the dynamic resource management problem in a timely fashion, as in-flight assets must be quickly re-tasked to respond to the changing environment. To facilitate the rapid generation of attractive updated mission plans, this procedure utilizes a technique for reducing the solution space, supports branching on multiple decision variables simultaneously, incorporates additional valid cuts to strengthen the minimal network constraints of the original mathematical model, and includes improved objective function bounds. An extensive numerical analysis indicates that the proposed approach significantly outperforms traditional branch-and-bound methodologies, and is capable of providing improved feasible solutions in a limited time. Although inspired by the dynamic resource management problem in particular, this approach promises to be an effective tool for solving other general types of vehicle routing problems.

Keywords: UAV routing; dynamic retasking; MILP; multiobjective; course of action; DVRPTW; branch-and-bound

1 Introduction

Unmanned aerial vehicles (UAVs) have become invaluable assets in global military operations. Technological advances and commander confidence in these systems have led to increasingly complex missions involving coordination among assets. For example, laser-guided munitions may require one UAV to 'paint' a target with a laser marker while another UAV fires the missile. Other complex mission components may involve battle damage assessment, surveillance of a target from multiple vantage points, and geolocation of unknown targets. Outside of the military domain, UAVs promise to be useful in search-and-rescue operations [24], disaster relief [19], police patrols [2], and security at major spectator events such as the Olympics [26].

Motivated by the enhanced capabilities of unmanned aircraft systems, as well as the fact that these aircraft operate in uncertain and rapidly evolving environments, the authors recently proposed a general modeling framework for dynamic re-routing of airborne assets [17]. Although inspired by unmanned aerial vehicles (UAVs), this model is applicable to a wide variety of vehicles, including land-based or submersibles. The aforementioned paper featured an integer linear programming (ILP) formulation for the problem of determining updated assignments of UAVs to a set of tasks when a dynamic event triggers the necessity for altering the initial mission plan. Examples of dynamic events (or 'pop-ups') may include the appearance of a new target, the loss of a UAV, or changes to characteristics of previously-identified tasks. In this problem, a set of priority-based, time-sensitive tasks are identified at the time of the pop-up event. A fleet of heterogeneous airborne resources are available; each resource begins its route at its current location and must terminate its route at a base location prior to exhausting its fuel supply. If a resource has insufficient fuel, 'dummy' bases exist as alternate base (terminal) nodes, which may be used with a penalty. Each task is classified according to a taxonomy that identifies whether the task is required or optional, and whether the task requires coordination among multiple resources or other tasks. Such synchronization may include multiple resources performing the same task simultaneously, multiple resources performing separate tasks simultaneously, or precedence constraints among tasks. The model presented in [17] is thought to be among the first formulations to reflect such a large variety of the complex task types performed by modern UAVs. We are aware of no solution approaches for similar formulations that are designed to produce attractive feasible solutions quickly.

Due to the time-critical nature of this problem, rapidly-generated implementable (albeit not necessarily optimal) solutions are required. Although the use cases presented in [17] indicate that many small-sized problems may be solved to optimality via commercial ILP solvers (e.g., CPLEX), such off-the-shelf solution approaches are ill-suited for finding quality solutions to larger problems in a reasonable time. In this paper, an 'intelligent' branch-and-bound solution procedure using the LP relaxation is proposed for solving this dynamic resource reallocation problem. This approach begins by employing a solution space reduction procedure that limits the number of candidate arcs in the network by taking into consideration the proximity of tasks to the initial route of each resource. Empirical analysis indicates that this procedure effectively reduces runtimes, a crucial feature for time-critical problems of this nature. Once the solution space has been reduced to a more manageable scale, a route construction procedure is initiated, whereby each resource begins its route at its current location and each subsequent arc is appended to the route until the resource is assigned to a base (depot) node.

Two rules are proposed for determining the decision variable, or sum of decision variables, on which the procedure branches. These rules exploit the underlying route construction procedure. As such, the determination of decision variables that are set during the branching process is guided by the last known node in each resource's route and the earliest feasible time for which the resource could begin service at the next node. If the bounding process reveals that a partially-constructed route cannot yield a better objective function value than the incumbent solution (which is initially provided by the original mission plan), the branching procedure is used to prohibit the resource from being assigned to the most recently assigned arc. The assignment of the chosen decision variable during the branching procedure need not be restricted to those decision variables that have non-zero values in the LP relaxation. In addition to employing an innovative method for determining branching policies, additional valid cuts are added when arc assignments are made. These cuts serve to strengthen the network formulation and effectively guide the resource to the next leg in its route.

Although our problem of interest was inspired by military UAV missions, both the mathematical model and the proposed solution approach may be applied to a more general class of dynamic resource re-allocation problems involving coordination among multiple resources (vehicles). For example, surveillance tasks requiring multiple UAVs to investigate a target could just as easily be considered as customer service requests requiring multiple repairmen to fix machinery. Similarly, a "battle damage assessment" task may be used to model precedence relationships requiring one type of good to be delivered before another good. Likewise, the expendable payload considered in this paper could describe bombs dropped from capacity-constrained UAVs or packages delivered by capacity-constrained trucks. It bears noting that, while some of these individual tasks may be considered by existing solution approaches, the problem of interest involves the combination of multiple complex task types. To solve this increasingly important problem, a new approach is required.

The remainder of this paper is organized as follows. In Section 2, a discussion of relevant solution approaches from the literature is presented. Section 3 describes the mathematical model and notation. An overview of our proposed solution approach is contained in Section 4, while the detailed approach is covered in Section 5. A numerical analysis is described in Section 6, including a comparison of our solution approach against CPLEX. Finally, a summary, discussion of the proposed approach's application to general vehicle routing problems, and suggested opportunities for further algorithmic research are presented in Section 7.

2 Solution Approaches for Related Problems

As discussed in the literature review of [17], there are numerous vehicle routing formulations that consider a subset of the features incorporated in our model. More recently, [9] has provided a survey of general vehicle routing problems employing a wide array of synchronization constraints. However, despite the growing interest in such coordinated vehicle routing problems, models that address the problem at hand are scarce.

One such model is presented by [14], who employ the LTL$_{-X}$ linear temporal logic language to translate a wide variety of mission constraints into two mixed integer linear programming (MILP) formulations; one based on set covering, the other on network flows. Examples involving up to four UAVs and three targets demonstrate several complex scenarios that may be modeled via this approach. These models are then solved with commercial ILP software. The LTL$_{-X}$ modeling approach is extended in [13] to address timing (rather than simply precedence) constraints among tasks. The resulting VRP with metric temporal logic (VRPMTL) is demonstrated on two highly-constrained problems involving three UAVs and five targets. The modeling approaches of [14] and [13] accommodate the description of complex tasks beyond those defined in [17]. However, no corresponding solution approaches for large-scale problems are provided.

Another similar problem is solved by the "coupled-constraint consensus-based bundle algorithm" (CCBBA) proposed by [28]. This approach is an extension of other bid-based approaches designed for decentralized mission planning and considers a variety of coordinated tasks with time windows as performed by vehicle of varying capabilities. Although the CCBBA approach is quite flexible, it is not capable of addressing all of our constraints in its present form. For example, it does not consider expendable payload or required tasks (a reward is earned when performing a task, but no penalties are assessed for failing to perform certain tasks). Also, some of the tasks described in our model would require a of nesting of CCBBA "activities," which is not allowed by that approach. Nonetheless, CCBBA is an intriguing approach that, if properly modified, may offer an effective means of solving our problem of interest.

Although we are aware of no documented solution approaches that have been tailored to solve realistically-sized instances of the problem at hand, it is worth noting the types of solution approaches that have been proposed for models that share some similarities with our work. In particular, we focus on UAV-related models because they are in our domain, and they (typically) recognize the importance of quick-running solution approaches due to the nature of this problem.

An exact solution approach to the problem of routing UAVs to targets with precedence constraints has been proposed by [5]. Their column generation procedure, which may be solved in a distributed manner, was demonstrated on a precedence-constrained problem involving four heterogeneous UAVs and six tasks. Other exact solution methodologies have been relegated to employing third-party mixed integer linear programming (MILP) solvers. Computational experiments for these methods involved a small number of targets. For example, the largest problem size considered by Weinstein and Schumacher [27] is 8 targets and 4 resources, and the largest size for Schumacher et al. [18] is 3 targets (where each target has 3 tasks) and 5 resources. Wilde et al. [29] do not provide any numerical analysis, but report that their approach has been applied to several scenarios involving 2 to 6 resources.

A heuristic solution approach is provided by Janez [12], who uses a modification of the well-known Clarke and Wright [6] savings algorithm. Bellingham et al. [1] first estimate completion times for each task from an enumerated list of all task combinations. An MILP is subsequently solved to determine which assignment to use for each resource. Finally, actual flight routes, subject to kinematic constraints regarding flight dynamics and collision avoidance, are determined. The largest test case considered is for 12 waypoints and 6 resources, with no discussion on solution times. More recently, [11] have proposed a tabu search heuristic with 2-opt exchange for a problem with precedence constraints. A simulation study involving two vehicles and 30 tasks was conducted, such that all tasks are performed instantaneously and vehicles are not bound by fuel restrictions. A survey of the dynamic vehicle routing (DVR) problem, in which new tasks are revealed over time, is provided by [4]. An "algorithmic queueing theory" approach is used to establish policies that guide future route determinations. Tasks are partitioned into clusters, such that tasks in a particular cluster are assigned to the same vehicle.

As with research on the vehicle routing problem with time windows (VRPTW), most recent solution approaches for UAV routing problems have focused on metaheuristics. A genetic algorithm is proposed by Shima et al. [21]. However, it is difficult to judge the applicability of their approach to our model, as their problem does not consider task priorities, time windows, payload delivery, or task coordination. Their algorithm was tested on 3 to 10 targets, where each target has exactly 3 tasks. A hybrid genetic approach is proposed by Berger et al. [3], who allow 15-minutes of runtime for their test cases. Shetty et al. [20] apply tabu search to test problems with up to 15 targets and 4 resources. Another tabu search heuristic is given by Kinney et al. [15], where the best move over all possible neighboring solutions is selected. They tested their heuristic on Solomon's VRP benchmark problems [22], which do not include time windows, heterogeneous resources, multiple depots, or task priorities, and require exactly one resource to perform each task. Additional testing on 10 multiple-depot VRP test problems was also conducted. The conclusion was that their heuristic was not able to produce solutions comparable to the best results reported for these benchmark problems, but attractive performance was found with significantly shorter runtimes. For problems involving multiple objectives, Tavana et al. [25] propose a method for evaluating candidate solutions. This approach may be included within a variety of metaheurstics, such as genetic algorithms, simulated annealing, and tabu search. Cordeau et al. [7] caution that, while metaheuristics have been used to find high-quality solutions for VRPTW problems, it is difficult to find feasible neighboring solutions when extensions such as multiple depots or heterogeneous vehicles are included.

3 Mathematical Model and Notation

This section is devoted to the mathematical formulation, details of which may be found in [17]. A summary of model notation regarding sets, parameters, and decision variables may be found in Tables 1, 2, and 3, respectively. The reader may note that, for ease of understanding, we have presented a slightly modified version of our original model.

The set M represents all known tasks, such that each task $j \in M$ is classified as being exactly one of five unique non-preemptive task types: RNA, ONA, RNS, $RNM(s)$, or RNP. The taxonomy defining these task type acronyms is described in detail in [17]. The first letter indicates whether the task is *required* (i.e., failure to perform this task results in an objective function penalty) or *optional*. The second letter indicates that these tasks are *non-preemptive*, such that these tasks may only be interrupted in the event of a pop-up. The third letter describes the unique characteristic of a task type. For example, an RNA task may be performed at *any* time within its allowable time window (to be described below). Each task $j \in RNA$ requires the assignment of at least $n_j^{\min}$, and no more than $n_j^{\max}$ resources. ONA tasks are the "optional" counterparts to RNA tasks, such that there is no penalty associated with failing to perform an ONA task. The RNS task type describes an individual required, non-preemptive task that must be performed by multiple resources *simultaneously* (although each distinct RNS task may be performed at a different time). The RNS task type may be useful for a target requiring direct overhead surveillance by multiple resources at the same time.

Table 1: Notation – Sets

B	Set of bases (depots).
M	Set of nonpreemptive tasks (mission components).
R	Set of resources.
B_r	Set of bases (depots) that are available to resource $r \in R$. $B_r \subseteq B$.
B'_r	If Δ_r^0 does not represent a base location, then $B'_r = \varnothing$. Otherwise, B'_r equals the actual base number associated with the initial location of resource $r \in R$.
M_r	Set of tasks capable of being performed by resource $r \in R$. $M_r \subseteq M$.
R_j	Set of resources that can perform task $j \in M$. $R_j \subseteq R$.
P	Set of tasks requiring payload delivery.
T	Set of time intervals comprising the mission horizon.
T_j	Set of time intervals in which service at node $j \in \{M \cup B \cup \Delta^*\}$ may begin. $T_j \subseteq T$.
Δ_r^0	Initial location of resource $r \in R$. This is a single location, not a set. $\Delta_r^0 \notin \{M \cup B \cup \Delta_r^*\}$. Resources cannot travel *to* Δ_r^0 ($\Delta_r^0 \notin \Delta_r^+$).
Δ_r^*	Sink node used by resource r if it does not have sufficient fuel capacity to travel to a base $b \in B_r$ prior to running out of fuel. $\Delta_r^* \notin \{M \cup B \cup \Delta_r^0\}$.
Δ_r^+	Set of all possible nodes to which resource $r \in R$ may travel. $\Delta_r^+ \subseteq \{M_r \cup B_r \cup \Delta_r^*\}$.
$\Delta_{r,j}^-$	Set of all possible nodes from which resource $r \in R$ may travel directly to node $j \in \Delta_r^+$. $\Delta_{r,j}^- \subseteq \{\Delta_r^0 \cup M\}$. $j \notin \Delta_{r,j}^-$.
RNA	Required, non-preemptive tasks that may be performed in any allowable time interval.
ONA	Optional, non-preemptive tasks that may be performed in any allowable time interval.
RNS	Required, non-preemptive tasks that must be performed by multiple resources simultaneously.
$RNM(s)$	Set of required, non-preemptive tasks in which multiple tasks must be performed simultaneously, where $s \in S$ is an index identifying a collection of related tasks.
RNP	Required, non-preemptive tasks with precedence requirements.
$RNP'(j)$	Set of tasks that are predecessors of task $j \in RNP$.

Table 2: Notation – Parameters

t_0	Time interval in which pop-up event occurs. $t_0 \equiv \min(T)$.
p_j	Priority value of task $j \in M$. $p_j > 0$.
$e_{r,j}$	Effectiveness value for resource $r \in R$ performing task $j \in M$.
d_j	Duration, in number of time intervals, for performing task $j \in M$.
$f_{r,i,j}$	Minimum number of time intervals required for resource $r \in R$ to travel to node $j \in \Delta_r^+$ from node $i \in \Delta_{r,j}^-$.
g'_r	Remaining fuel capacity of resource r at time t_0. Measured in number of time intervals.
$n_j^{\min}(n_j^{\max})$	Minimum (maximum) number of resources that may perform task $j \in M$.
$k_j^{\min}(k_j^{\max})$	Minimum (maximum) units of payload that may be delivered to task $j \in P$.
c'_r	Remaining payload capacity of resource r at time t_0.
α	Objective function scaling parameter used to capture the relative importance of maximizing overall mission effectiveness ($\alpha = 1$) or minimizing changes to the initial mission plan ($\alpha = 0$). $0 \leq \alpha \leq 1$.
β	Objective function scaling parameter to capture the importance of minimizing total travel distance. $\beta \geq 0$.
$a_{r,i,j}$	Binary parameter. If resource r were initially assigned to travel directly from node i to node j then $a_{r,i,j} = 1$. Let $a_{r,\bullet,j} = 1$ if resource r were initially assigned to node j (regardless of the node preceding j).
lag_j	Number of time intervals that must pass between the time that the last task in $RNP'(j)$ is performed and the beginning of service of task $j \in RNP$.

Table 3: Notation – Decision Variables

$x^t_{r,i,j}$	Binary decision variable. $x^t_{r,i,j} = 1$ if resource $r \in R$ is to be assigned to node $j \in \Delta^+_r$ during time interval $t \in T_j$ and the previous node was $i \in \Delta^-_{r,j}$.
$x^{\text{RNP}}_{t,j}$	Binary decision variable used to indicate if task $j \in RNP$ is performed during time interval $t \in T_j$.
$x^{\text{RNP}'}_j$	Binary decision variable used to indicate if at least one task in $RNP'(j)$ is performed, for all $j \in RNP$.
y_j	Infinite resource. $y_j \in \{0, 1, \ldots, n^{\min}_j\}$ for all $j \in \{RNA \cup RNP\}$.
z^t_j	Infinite resource. $z^t_j \in \{0, 1, \ldots, n^{\min}_j\}$ for all $j \in \{RNS \cup RNM(s)\}$, $t \in T_j$.
w_j	Infinite payload. $w_j \in \{0, 1, \ldots, k^{\min}_j\}$ for all $j \in P$.
$q_{r,j}$	Integer decision variable used in payload delivery constraints. Represents the units of payload delivered by resource r to task (target) $j \in P$. $q_{r,j} \in \{0, 1, \ldots, \min(c'_r, k^{\max}_j)\}$.
h^t_j	Binary decision variable used by RNS tasks, such that $h^t_j = 1$ if task j is performed in time interval $t \in T_j$.
b^t_s	Binary decision variable used by RNM tasks to ensure that all tasks in the s^{th} set, $RNM(s)$, are performed in the same time interval, t.

$RNM(s)$ tasks represent an extension of RNS tasks, whereby *multiple* tasks must be performed simultaneously. Here, $s \in S$ is an index allowing the definition of multiple sets of $RNM(s)$ tasks. Thus, for each s there exists a set of multiple tasks that must be performed simultaneously. $RNM(s)$ tasks may be useful for coordinating laser-guided munitions, where one resource may use a laser from close range to designate a target while another resource may fire the weapon from a different location. Finally, RNP tasks incorporate *precedence* constraints, such that task $j \in RNP$ must be performed after task $k \in RNP'(j)$ is completed, where $RNP'(j)$ represents the set of all predecessors of task j. The parameter lag_j indicates the minimum number of time intervals that must pass between the completion of predecessor task k and the start of task j.

Associated with each task $j \in M$ is a time window, T_j, that represents the set of discrete time intervals in which the task may be performed. Each task j requires a known performance duration, such that $d_j = 0$ indicates an instantaneous task (e.g., taking a snapshot) and $d_j > 0$ indicates a task requiring multiple time slices to complete (e.g., video surveillance). The relative priority of task j is given by p_j, where larger values of p_j represent higher-priority tasks. To reflect the fact that the configuration of resource r will affect that resource's ability to perform task j, the parameter $e_{r,j}$ denotes the relative effectiveness of r while performing task j. Some tasks may require the delivery of expendable payload carried by one or more resources, such as laser-guided missiles.

The key binary decision variable $x^t_{r,i,j}$ serves to describe each leg in the route of each resource, where $x^t_{r,i,j} = 1$ if resource $r \in R$ begins service at node j at time $t \in T_j$, after having previously visited node i. The concepts of 'infinite' resources (IR), 'infinite' payload (IP), and 'dummy' bases (DB) were proposed to ensure that a mathematically feasible solution can always be found. These are analogous to artificial variables with a high cost, allowing for feasible/partial and implementable solutions.

Due to the complexity of the model presented in [17], some simplifying assumptions are utilized in this paper. First, the solution procedure proposed here focuses solely on the five non-preemptive task types (RNA, ONA, RNS, $RNM(s)$, and RNP). Although the four additional preemptive task types in [17] are attractive due to their flexibility, they require significantly more decision variables. Fortunately, any solution in which a preemptive task is not preempted is still a valid solution. Therefore, although the flexibility of these tasks is lost, they may be included in the proposed solution approach if they are treated as non-preemptive tasks. Second, this solution approach relies on assigning resources to tasks at the earliest feasible time. As such, this approach does not account for time preferences within each task's allowable time window. Furthermore, the objective of minimizing changes to the *time* in which a given task is performed is not incorporated in the branching rules.

3.1 Objective Function

The objectives of the dynamic resource reallocation problem are to maximize the overall effectiveness of the mission, minimize changes to the initial mission plan, minimize total travel time, and minimize the use of IR, IP, and ’dummy’ bases. Overall mission effectiveness is quantified by the task priorities and the relative effectiveness of the resources assigned to those tasks. The objective of minimizing mission changes is considered to prevent an inconsequential improvement in mission effectiveness at the expense of wholesale changes in task assignments. As will be discussed in Section 5, greater emphasis on minimizing changes can lead to improved algorithm performance.

The objectives are combined in a linear manner as

$$\text{Max } \alpha \frac{Z_E}{Z_E^{\max}} - (1-\alpha)\left(\frac{Z_{CO}}{Z_{CO}^{\max}} + \frac{Z_{CR}}{Z_{CR}^{\max}}\right) - \beta \frac{Z_T}{Z_T^{\max}} - \left(Z_{IR} + Z_{DB} + Z_{IP}\right), \tag{1}$$

where $0 \leq \alpha \leq 1$ captures the tradeoff between maximizing effectiveness and minimizing changes, while $0 \leq \beta \leq 1$ reflects the weight assigned to minimizing travel time. The $Z_\bullet$ terms represent corresponding objectives, such that Z_E describes the effectiveness of assignments; Z_{CO} and Z_{CR} represent changes to the order in which resources perform tasks and the resources that are assigned to each task, respectively; Z_T represents the total travel time; and Z_{IR}, Z_{IP} and Z_{DB} represent the use of IR, IP, and ‘dummy’ bases. Expressions for these terms are provided below:

$$Z_E \equiv \sum_{j \in M} \sum_{r \in R_j} \sum_{t \in T_j} \sum_{i \in \Delta^-_{r,j}} p_j e_{r,j} \left(1 - \left(\frac{t - \min\{T_j\}}{|T_j| + 1}\right)\right) x^t_{r,i,j} + \sum_{j \in P} \sum_{r \in R_j} p_j q_{r,j},$$

$$Z_E \leq Z_E^{\max} \equiv \sum_{j \in M} \left(p_j \max_{r \in R_j} \left(e_{r,j}\right) n_j^{\max} + p_j k_j^{\max}\right),$$

$$Z_{CO} \equiv \sum_{r \in R} \sum_{j \in \Delta^+_r} \sum_{\substack{i \in \Delta^-_{r,j} \\ i \neq j}} \sum_{t \in T_j} \left(1 - a_{r,i,j}\right) x^t_{r,i,j} \leq Z_{CO}^{\max} \equiv \sum_{j \in M} n_j^{\max} + |R|,$$

$$Z_{CR}^{\text{non}} \equiv \sum_{j \in M^{\text{non}}} \sum_{r \in R_j} a_{r,\bullet,j} \left(1 - \sum_{i \in \Delta^-_{r,j}} \sum_{t \in T_j} x^t_{r,i,j}\right),$$

$$Z_{CR}^{\text{base}} \equiv \sum_{r \in R} \sum_{j \in B_r} a_{r,\bullet,j} \left(1 - \sum_{i \in \Delta^-_{r,j}} \sum_{t \in T_j} x^t_{r,i,j}\right),$$

$$Z_{CR} \equiv Z_{CR}^{\text{non}} + Z_{CR}^{\text{base}} \leq Z_{CR}^{\max} \equiv \sum_{j \in M} n_j^{\max} + |R|,$$

$$Z_T \equiv \sum_{r \in R} \sum_{j \in \Delta^+_r} \sum_{i \in \Delta^-_{r,j}} \sum_{t \in T_j} f_{r,i,j} x^t_{r,i,j} \leq Z_T^{\max} \equiv \sum_{r \in R} g'_r,$$

$$Z_{IR} \equiv \sum_{j \in (RNA \cup RNP)} p_j y_j + \sum_{j \in (RNS \cup RNM(s))} \sum_{t \in T_j} p_j z^t_j,$$

$$Z_{DB} \equiv \sum_{r \in R} \sum_{i \in \Delta^-_{r,\Delta^*_r}} x^{t_0+g'_r+1}_{r,i,\Delta^*_r},$$

$$Z_{IP} \equiv \sum_{j \in P} p_j w_j.$$

3.2 Task Constraints

The following constraints ensure that each task is properly assigned, according to its type. First, each RNA task must be performed by the appropriate number of resources at any time within an allowable time window. Additionally, a given resource may perform a particular RNA task no more than once.

$$n^{\min}_j \leq \sum_{r \in R_j} \sum_{t \in T_j} \sum_{i \in \Delta^-_{r,j}} x^t_{r,i,j} + y_j \leq n^{\max}_j \qquad \forall\ j \in RNA, \tag{2}$$

$$\sum_{t \in T_j} \sum_{i \in \Delta^-_{r,j}} x^t_{r,i,j} \leq 1 \qquad \forall\ j \in \{RNA : n^{\max}_j \geq 2\}, r \in R_j. \tag{3}$$

Each optional task $j \in ONA$ may be performed by at most $n^{\max}_j$ resources. As with RNA tasks, an ONA task may not be performed more than once by the same resource.

$$\sum_{r \in R_j} \sum_{t \in T_j} \sum_{i \in \Delta^-_{r,j}} x^t_{r,i,j} \leq n^{\max}_j \qquad \forall\ j \in ONA, \tag{4}$$

$$\sum_{t \in T_j} \sum_{i \in \Delta^-_{r,j}} x^t_{r,i,j} \leq 1 \qquad \forall\ j \in \{ONA : n^{\max}_j \geq 2\}, r \in R_j. \tag{5}$$

A particular task $j \in RNS$ must be performed by an acceptable number of resources *simultaneously*:

$$h^t_j n^{\min}_j \leq \sum_{r \in R_j} \sum_{i \in \Delta^-_{r,j}} x^t_{r,i,j} + z^t_j \leq h^t_j n^{\max}_j \qquad \forall\ j \in RNS, t \in T_j, \tag{6}$$

$$\sum_{t \in T_j} h^t_j = 1 \qquad \forall\ j \in RNS. \tag{7}$$

Similarly, the multiple unique tasks within set $RNM(s)$, for a particular index $s \in S$, must be performed at the same time:

$$b^t_s n^{\min}_j \leq \sum_{r \in R_j} \sum_{i \in \Delta^-_{r,j}} x^t_{r,i,j} + z^t_j \leq b^t_s n^{\max}_j \qquad \forall\ s \in S, j \in RNM(s), t \in \bigcap_{k \in RNM(s)} T_k, \tag{8}$$

$$\sum_{t \in \bigcap_{k \in RNM(s)} T_k} b^t_s = 1 \qquad \forall\ s \in S. \tag{9}$$

Finally, each task $j \in RNP$ must be performed if at least one task $k \in RNP'(j)$ is previously performed. In Constraint (10) below, $x^{\mathrm{RNP'}}_j = 1$ if a predecessor task k is performed. Constraint (11) ensures that the appropriate number of resources are assigned to task j, provided that predecessor task k is performed. As with the other task types, an RNP task cannot be performed multiple times by the same resource, as in Constraint (12). Constraint (13) determines the start time of task j, as indicated by $x^{textRNP}_{t,j}$. Constraint (14) prohibits RNP task j from starting before lag_j

time intervals have elapsed since the completion of predecessor task k.

$$x_j^{\mathrm{RNP}'} \leq \sum_{k\in RNP'(j)} \sum_{r\in R_k} \sum_{i\in\Delta_{r,k}^-} \sum_{t\in T_k} x_{r,i,k}^t \leq \left(\sum_{k\in RNP'(j)} n_k^{\max} \right) x_j^{\mathrm{RNP}'} \qquad \forall\ j\in RNP, \tag{10}$$

$$n_j^{\min} x_j^{\mathrm{RNP}'} \leq \sum_{r\in R_j} \sum_{i\in\Delta_{r,j}^-} \sum_{t\in T_j} x_{r,i,j}^t + y_j \leq n_j^{\max} x_j^{\mathrm{RNP}'} \qquad \forall\ j\in RNP, \tag{11}$$

$$\sum_{t\in T_j} \sum_{i\in\Delta_{r,j}^-} x_{r,i,j}^t \leq 1 \qquad \forall\ j\in \{RNP : n_j^{\max} \geq 2\}, r\in R_j, \tag{12}$$

$$x_{t,j}^{\mathrm{RNP}} \leq \sum_{r\in R_j} \sum_{i\in\Delta_{r,j}^-} x_{r,i,j}^t \leq n_j^{\max} x_{t,j}^{\mathrm{RNP}} \qquad \forall\ j\in RNP, t\in T_j, \tag{13}$$

$$\left(1 - x_{t,j}^{\mathrm{RNP}}\right) \geq \frac{1}{n_k^{\max}} \sum_{r\in R_k} \sum_{i\in\Delta_{r,k}^-} \sum_{\substack{t_k\in T_k: \\ t_k > t - lag_j - d_k}} x_{r,i,k}^{t_k} \qquad \forall\ j\in RNP, k\in RNP'(j), t\in T_j. \tag{14}$$

3.3 Payload Delivery Constraints

Constraints governing the allocation of expendable payload are as follows:

$$\sum_{j\in\{P\cap M_r\}} q_{r,j} \leq c_r' \qquad \forall\ r\in R, \tag{15}$$

$$q_{r,j} \leq \min(c_r', k_j^{\max}) \sum_{i\in\Delta_{r,j}^-} \sum_{t\in T_j} x_{r,i,j}^t \qquad \forall\ j\in P, r\in R_j, \tag{16}$$

$$k_j^{\min} \leq \sum_{r\in R_j} q_{r,j} + w_j \leq k_j^{\max} \qquad \forall\ j\in P, \tag{17}$$

3.4 Network Constraints

The following constraints ensure that each resource is assigned to a feasible route.

$$\sum_{j\in\{\Delta_r^+ : t\in T_j\}} \sum_{i\in\Delta_{r,j}^-} x_{r,i,j}^t \leq 1 \qquad \forall\ r\in R, t\in T, \tag{18}$$

$$\sum_{j\in\{\Delta_r^+ : \Delta_r^0\in\Delta_{r,j}^-\}} \sum_{t\in T_j} x_{r,\Delta_r^0,j}^t = 1 \qquad \forall\ r\in R, \tag{19}$$

$$\sum_{t\in\{T_j : t< t_0 + f_{r,\Delta_r^0,j}\}} x_{r,\Delta^0,j}^t = 0 \qquad \forall\ r\in R, j\in\{\Delta_r^+ : \Delta_r^0\in\Delta_{r,j}^-\}, \tag{20}$$

$$\sum_{j\in B_r} \sum_{i\in\Delta_{r,j}^-} \sum_{t\in\{T_j : t\leq t_0+g_r'\}} x_{r,i,j}^t + \sum_{i\in\Delta_{r,\Delta_r^*}^-} x_{r,i,\Delta_r^*}^{t_0+g_r'+1} = 1 \qquad \forall\ r\in\{R : B_r' = \varnothing\}, \tag{21}$$

$$\begin{aligned} t_0 + 1 &\leq \sum_{j\in B_r} \sum_{i\in\Delta_{r,j}^-} \sum_{t\in T_j} t x_{r,i,j}^t + \sum_{i\in\Delta_{r,\Delta_r^*}^-} (t_0 + g_r') x_{r,i,\Delta_r^*}^{t_0+g_r'+1} \\ &\leq \sum_{j\in\Delta_r^+} \sum_{t\in T_j} (t - f_{r,\Delta_r^0,j} + g_r') x_{r,\Delta_r^0,j}^t \qquad \forall\ r\in\{R : B_r' \neq \varnothing\}, \end{aligned} \tag{22}$$

$$\sum_{i\in\Delta^-_{r,j}} \sum_{t\in T_j} x^t_{r,i,j} = \sum_{k\in\{\Delta^+_r : j\in\Delta^-_{r,k}\}} \sum_{t\in T_k} x^t_{r,j,k} \qquad \forall\ r\in R, j\in M, \tag{23}$$

$$\sum_{i\in\Delta^-_{r,j}} \sum_{t\in T_j} t x^t_{r,i,j} \leq \sum_{k\in\{\Delta^+_r : j\in\Delta^-_{r,k}\}} \sum_{t\in T_k} (t - f_{r,j,k}) x^t_{r,j,k} \qquad \forall\ r\in R, j\in M. \tag{24}$$

3.5 Decision Variable Definitions

$$x^t_{r,i,j} \in \{0,1\} \qquad \forall\ r\in R, j\in\Delta^+_r, i\in\Delta^-_{r,j}, t\in T_j, \tag{25}$$

$$y_j \in \{0,1,\ldots,n^{\min}_j\} \qquad \forall\ j\in\{RNA\cup RNP\}, \tag{26}$$

$$z^t_j \in \{0,1,\ldots,n^{\min}_j\}, h^t_j \in \{0,1\} \qquad \forall\ j\in RNS, t\in T_j, \tag{27}$$

$$z^t_j \in \{0,1,\ldots,n^{\min}_j\} \qquad \forall\ s\in S, j\in RNM(s), t\in \bigcap_{k\in RNM(s)} T_k, \tag{28}$$

$$b^t_s \in \{0,1\} \qquad \forall\ s\in S, t\in \bigcap_{k\in RNM(s)} T_k, \tag{29}$$

$$x^{\mathrm{RNP}'}_j \in \{0,1\} \qquad \forall\ j\in RNP, \tag{30}$$

$$x^{\mathrm{RNP}}_{t,j} \in \{0,1\} \qquad \forall\ j\in RNP, t\in T_j, \tag{31}$$

$$q_{r,j} \in \{0,1,\ldots,\min(c'_r, k^{\max}_j)\} \qquad \forall\ j\in P, r\in R_j, \tag{32}$$

$$w_j \in \{0,1,\ldots,k^{\min}_j\} \qquad \forall\ j\in P. \tag{33}$$

4 Overview of Solution Approach

This section describes the general features of the ‘intelligent’ branch-and-bound procedure. This procedure begins with a solution space reduction technique. Next, the initial mission plan serves as the incumbent solution, where each resource begins at its current location (at time t_0) and visits all tasks in the resource’s initial route that have yet to be completed. ‘Infinite’ resources (IR) are used to maintain feasbility, as pop-up tasks or tasks that were initially assigned to resources that have been taken out of service will not be assigned in the incumbent solution. Because IR are penalized, the incumbent objective function value may be a large negative number, depending upon the quantity of IR that are required to ensure a feasible incumbent solution.

The linear programming (LP) relaxation will likely result in many fractional arcs that would call for portions of each resource to be split across numerous arcs (or possibly along the same arc at different times). Our approach is to use a guided tree search (intelligent branch-and-bound) to build each resource’s path forward through the network, starting at the resource’s initial location and terminating at a base node prior to exhausting its fuel supply.

At any point in the route construction procedure, IR may be employed to establish a feasible (albeit heavily penalized) solution. As the branching process proceeds further down the branch-and-bound tree, the number of tasks assigned to each resource increases, the number of IR required decreases, and potentially good implementable solutions are generated. For this reason, a depth-first branching strategy is used. Aside from the obvious advantage of reducing the reliance upon IR more quickly, the depth-first strategy has lower memory requirements and is easier to implement than other strategies.

To simplify explanation of the branching process, the ‘left’ branches in the branch-and-bound tree correspond to the *assignment* of a resource to an arc, while the ‘right’ branches correspond to the *prohibition* of the resource from taking said arc at *any* time. The exact nature of each *assignment* is determined by the task type of the destination node in the candidate arc. Unlike standard branch-and-bound approaches, in which a non-zero decision variable is selected, our approach may select a decision variable with a value of zero in the LP relaxation. In this assignment branch, a selected resource is forced to travel a certain arc. Depending upon the task type, the assignment may be at the earliest feasible time, or it may be over a range of times. This concept is explained more fully later in this section.

As each arc is assigned, additional valid cuts are added to strengthen the network constraints. These cuts are motivated by the relative weakness of the network formulation. Although the network formulation is valid, it is designed to be minimal, thus requiring fewer constraints to define the still-sizable model. The advantage of this approach is that the model may be generated more quickly. The obvious drawback is that the LP relaxation is not as tight. However, by adding constraints on an as-needed basis, the network constraints are strengthened in a more efficient manner, thus reducing the number of redundant constraints.

In the remainder of this section, details regarding the solution space reduction, the determination of resource routes from the LP relaxation, rules designed to select assignments in the modified branch-and-bound procedure, and additional constraints included in the branching process are detailed.

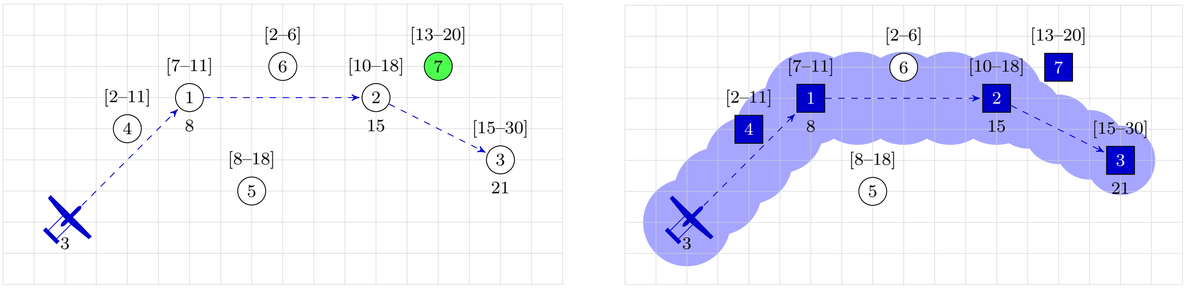
4.1 Reducing the Solution Space

To facilitate the rapid generation of attractive feasible solutions, a technique for reducing the size of the solution space is proposed. This approach seeks to eliminate certain binary $x^t_{r,i,j}$ decision variables, which characterize each leg of the route assigned to resource r, that are unlikely to appear in an optimal solution.

From the definition of $x^t_{r,i,j}$, the number of defined decision variables may be reduced by limiting the size of Δ^+_r, the set of nodes that may be visited by resource r. As in the definition, node $j \in B_r$ will remain in set Δ^+_r. To determine if task $j \in M$ is included in set Δ^+_r, first, let $j \in \Delta^+_r$ if r were initially assigned to perform task j (i.e., if parameter $a_{r,\bullet,j} = 1$). Next, if j is a pop-up task and $e_{r,j} > 0$, then let $j \in \Delta^+_r$. Finally, for each arc (i, k) in the initial mission plan of resource r (i.e., for all arcs (i, k) such that parameter $a_{r,i,k} = 1$), let task $j \in M$ be assigned to the set Δ^+_r if the following conditions are met: (1) $e_{r,j} > 0$ (resource r is capable of performing task j), (2) the allowable time window for task j overlaps the time windows for nodes i and k (i.e., $\min(T_j) \leq \max(T_k)$ and $\max T_j \geq \min(T_i)$), and (3) task j lies within a pre-determined distance from the arc between nodes i and k.

This last condition is applied by constructing overlapping circles of a pre-determined radius along each initially-assigned arc, where the first circle is centered at the location of node i and all subsequent circles are centered at the intersection of the preceeding circle with the arc. This facilitates construction of the region that is 'near' the arc, allowing rapid calculations to identify the nodes within this region. In this study, we state the radius of each circle in terms of the percentage of the distance between the two nodes comprising an arc, and denote this radius as $RadScale$.

An example of the solution space reduction approach is shown in Figure 1, where a pop-up task (node 7) appears at time 3. Bracketed numbers above each node represent the allowable time windows, while numbers below nodes represent the time at which r was initially assigned to begin service at that node. The reduced solution space is represented in Figure 1b, where node j is shown as a dark square if $j \in \Delta^+_r$. Node 5 is not included in Δ^+_r because it lies outside of the shaded area. Notice that, although node 6 lies within the shaded area, its time window does not overlap the windows for nodes 1 and 2.



(a) Initial assigned route for resource r. Tasks 4–6 were initially assigned to other resources.

(b) Δ^+_r is determined by the solution space reduction approach with $RadScale = 0.25$.

Figure 1: Application of the Partitioning Approach

Thus, the proposed solution space reduction approach takes into account the spatiotemporal aspects of the initial mission plan by including tasks that are 'near' the initial route and also have time windows that overlap the time windows of the nodes contained in the initial route. It should be noted that if $RadScale = 0$, only initially-assigned tasks and pop-ups are included in Δ^+_r; if $RadScale = \infty$, all tasks for which r is capable of performing are included in Δ^+_r. Our solution approach dynamically varies the size of $RadScale$.

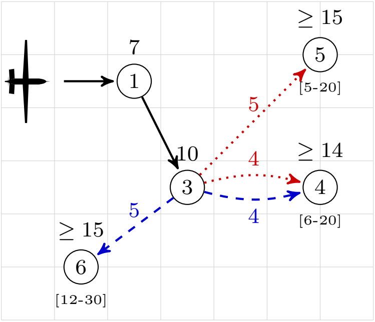
4.2 Extracting Resource Routes from the LP Relaxation

The solution to the LP relaxation may be used to trace the progress of each resource's route, from Δ^0_r to the last consecutive integer assignment. For example, if $x^{t_3}_{r,\Delta^0_r,3} = x^{t_2}_{r,3,2} = x^{t_5}_{r,2,5} = 1$, then resource r has a partially completed route beginning at Δ^0_r and terminating at node 5. More formally, let i_r represent the last assigned node in the route for resource r and let t_r represent the time associated with the beginning of service at node i_r. Because each resource r begins at node Δ^0_r at time slice t_0, we may initialize $i_r = \Delta^0_r$ and $t_r = t_0$. Next, if $x^t_{r,i_r,j} = 1$ for any $j \in \Delta^+_r$ and $t \in T_j$, then resource r is assigned to arc (i_r, j). Update the last known node for resource r by setting $i_r = j$ and $t_r = t$. If i_r is a base node, the route for resource r has been fully constructed from source to sink. Otherwise, continue this process until $x^t_{r,i_r,j} < 1$. At this point, there will be multiple decision variables emanating from node i_r with fractional values (i.e., $x^t_{r,i_r,j} < 1$ for some $j \in \Delta^+_r$ and at some time $t \in T_j$). Thus, the LP relaxation does not provide an explicit assignment for the next leg in the route of resource r, and a forced assignment must be made.

To assist in the determination of the forced assignment, these fractional arcs are classified according to whether a particular fractional arc appears (chronologically) before or after the last assigned arc in a resource's path. **'Out-of-Sequence' Arcs** are fractional arcs for resource r that emanate from node i_r and have an assigned time *prior to* t_r, where t_r represents the time at which node i_r is slated to begin service in the partially-constructed route. We define set OoS to be all (r, i_r, j, t) such that $0 < x^t_{r,i_r,j} < 1$ and $t < t_r$. Although it may be feasible to assign r to out-of-sequence arc (i_r, j), it is clearly not feasible to do so at time t. **'Sequenced' Arcs** are fractional arcs that emanate from node i_r and have an assigned time *after* t_r. We define set Seq to be all (r, i_r, j, t) such that $0 < x^t_{r,i_r,j} < 1$ and $t > t_r$. The Seq set identifies all non-zero decision variables $x^t_{r,i_r,j} < 1$ for which time t is after the time service begins at node i_r. If $t \geq t_r + f_{r,i_r,j}$, it may be feasible to assign r to arc (i_r, j) at time t. In the event that node j is a task that possesses timing constraints requiring multiple resources to begin service simultaneously, time t might not be a feasible start time unless all other resources assigned to j may also begin service at time t.

Note that the ILP formulation contains constraints that prohibit resources from multi-tasking, thus preventing the occurrence of $x^t_{r,j,k} > 0$ for any $t = t_r$.

An example of OoS and Seq fractional arcs for a single resource is presented in Figure 2. Here, previous iterations of the branching procedure have already constructed a partial route from Δ^0_r to task 1 to task 3, such that $i_r = 3$ and $t_r = 10$. The numbers above nodes 1 and 3 represent the times at which r is assigned to begin service at the respective tasks. The numbers above nodes 4 – 6 represent the earliest feasible time for which r could be assigned to these tasks, where travel times are shown along each arc. Finally, the closed interval of allowable time slices for each task is shown below the as-yet-unassigned nodes.



Decision Variable		Value	Arc Type
$x^5_{r,3,5}$	$=$	0.25	out-of-sequence (dotted)
$x^6_{r,3,4}$	$=$	0.20	out-of-sequence (dotted)
$x^7_{r,\Delta^0_r,1}$	$=$	1.00	constructed route (solid)
$x^{10}_{r,1,3}$	$=$	1.00	constructed route (solid)
$x^{12}_{r,3,6}$	$=$	0.20	sequenced (dashed)
$x^{13}_{r,3,4}$	$=$	0.35	sequenced (dashed)

Figure 2: An example of ‘out-of-sequence’ and ‘sequenced’ arc types.

4.3 Determining the Earliest Starting Time for the Next Destination

One of the key aspects of the route construction procedure is that resources are assigned to the next node at the earliest feasible time. The determination of this time, denoted as t^{early}, is task-type-specific. For RNA and ONA tasks, as well as travel to base (terminal) nodes, the determination of t^{early} is straightforward. Given that r begins service at node i_r at time t_r, the earliest feasible time to begin service at some node j' would be the minimum allowable time of node j that is not prior to $t_r + f_{r,i_r,j'}$, where $f_{r,i_r,j'}$ represents the minimum number of time intervals required for resource r to travel arc (i_r, j'). For RNP tasks, where resources must observe a lag time after performing task $k \in RNP'(j)$, the earliest feasible time assignment for r must account for the time window for task j, the travel time for the resource from i_r to j, as well as the required lag time after task k is completed. Because RNA, ONA, RNP, and base nodes do not require concurrent time coordination with other nodes, the assignment phase of the branching process (the ‘left’ branch) may be used to force resource r to travel arc (i_r, j) and begin service at time t^{early}. Thus, the assignment would be to set $x^{t^{\text{early}}}_{r,i_r,j} = 1$ for the appropriate resource r and destination j.

For tasks involving time-coordination considerations, additional logic is required to determine the earliest feasible starting time. For RNS tasks (where multiple resources must begin service at a single task simultaneously) and $RNM(s)$ tasks (where multiple resources must begin service at *multiple* tasks simultaneously), the earliest feasible time assignment for resource r must account for the time window for task j and the travel time for the resource from i_r to j. It must also account for the earliest feasible time that all *other* resources assigned to j could begin service. The determination of this time, denoted by t^{latest}_j, may be found by parsing the solution from the LP relaxation, such that $t^{\text{latest}}_j = \max_{r' \in R}\{\min(t \in T_j : x^t_{r',i_{r'},j} > 0)\}$ This is the latest time that task $j \in \{RNS \cup RNM(s)\}$ has been assigned, and it represents the earliest possible time that any resource could be assigned to coordinated task j.

Due to the time-coordination requirements that must be observed by multiple resources, the assignment phase of the branching procedure must not blindly force a given resource to begin service at task $j \in \{RNS \cup RNM(s)\}$ at the earliest feasible time. Instead, the assignment should first force the resource to travel arc (i_r, j) and begin service *on or after* t^{early}. Next, the assignment of the specific time for which service at task j is to begin may be set for *all* resources only after it is clear that no other resources could be assigned to j. Let $t'' = t^{\text{early}}$ if a resource has not been assigned to a particular arc (i_r, j), and $t'' = \max(T_j)$ if a resource has already been assigned to (i_r, j). The value of t'' is used to determine if the branching process should assign a resource to arc (i_r, j) at some time on or after t^{early}, or if it should assign all resources to begin servicing j at time t^{early}. We will assign all resources that have been routed along arc (i_r, j) to begin service at task j only if all other resources have been assigned to tasks that begin after $\max(T_j)$.

Because the values of t^{early} and t'' are used extensively in our two proposed branching rules, the following function, `findEarlyStart`, is defined. This function returns t^{early} and t'' for a particular candidate assignment of resource r to some arc (i_r, j).

```
**Function** findEarlyStart($r, i_r, t_r, j$):
  $t'' = t^{\text{early}} = \min(t \in T_j : t \geq t_r + f_{r,i_r,j})$
  $t_j^{\text{latest}} = \max_{r' \in R}\{\min(t \in T_j : x^{t}_{r',i_{r'},j} > 0)\}$
  **if** $j \in (RNS \cup RNM(s))$ **then**
    $t^{\text{early}} = \max(t^{\text{early}}, t_j^{\text{latest}})$
    **if** $\sum_{t_j \in T_j : t_j \geq t_j^{\text{latest}}} x^{t_j}_{r,i_r,j} = 1$ **then**
      $t'' = \max(T_j)$ [$r$ assigned to task $j$.]
    **else**
      $t'' = t^{\text{early}}$
    **end if**
  **end if**
  **if** $j \in RNP$ **then**
    $t^{RNP'} = \max_{r' \in R, k \in RNP'(j), t_k \in T_k} (d_k + t_k : x^{t_k}_{r',i_{r'},k} > 0)$
    $t^{\text{early}} = \min(t \in T_j : t \geq t_r + f_{r,i_r,j} \text{ and } t \geq t^{RNP'} + lag_j)$
    $t'' = t^{\text{early}}$
  **end if**
  **return** $t^{\text{early}}$ and $t''$
```

4.4 Selecting the Assignment

After solving the LP relaxation and extracting the partially-constructed route for each resource, the branching process requires guidance in selecting the next assignment of a resource r to some arc (i_r, j). Two rules for determining this assignment are proposed, and are identified as Rules A1 and C1. Three other rules were explored in [16], but for the sake of brevity are omitted from this discussion. Both branching rules determine a resource (r'), a next destination node (j'), and the corresponding service start time (t').

4.4.1 Rule A1

Rule A1 first attempts to select the assignment of a resource based on the OoS arcs. The rationale is that these arcs appear out-of-sequence because subsequent nodes in the route require these tasks to be performed early. As a result, although the times assigned to these *OoS* arcs are infeasible in the LP relaxation, it might be helpful to assign these nodes as early as possible.

If no feasible assignments for *OoS* arcs were found, the rule attempts to make the assignment based on sequenced fractional arcs that lead to *task* nodes. Here, we look at each arc in the order of time in the decision variable subscripts. Thus, preference is given to nodes that are assigned early in the fractional solution. Arcs leading to bases are considered only if no feasible assignment to a task node can be found. This is motivated by the fact that bases represent sink nodes in the network, such that resources may not return to service after visiting a base. In this manner, Rule A1 attempts to delay the assignment of resources to bases.

Finally, if neither *OoS* nor *Seq* arcs provide feasible assignments, the *Seq* arc with the highest fractional decision variable value is chosen. As this will result in an infeasible assignment, it is unnecessary to solve the LP relaxation after making this assignment. Instead, the 'right' branch prohibiting the infeasible arc may be taken, where (hopefully) a fractional arc leading to a time-feasible destination will be produced in the resulting LP relaxation. Note that Rule A1 does not exclude any potentially-valid solutions from consideration. As such, Rule A1 is not a heuristic approach for a given partition of the solution space, provided that runtimes are sufficiently long to allow the branching process to fathom all nodes in the branch-and-bound tree. Rule A1 may be expressed as follows:

Step 0 – Initialization:
Set $r' = j' = t' = \varnothing$ (No assignment has been determined).
Set $t^{\text{asgn}} = \max(T) + 1$ (Initialize to a late time).

Step 1 – From the set of all out-of-sequence fractional arcs, find the destination that may be feasibly started the earliest. Break ties by selecting resources in ascending order.

```
for all $(r, i_r, j, t) \in OoS$ do
  $(t^{\text{early}}, t'') = \texttt{findEarlyStart}(r, i_r, t_r, j)$
  if $(t'' < t^{\text{asgn}})$ then
    $r' \leftarrow r,\ j' \leftarrow j,\ t' \leftarrow t^{\text{early}},\ t^{\text{asgn}} \leftarrow t''$
  end if
end for
if no feasible assignment was found then go to Step 2. Else, stop, returning the values for $r'$, $j'$, and $t'$.
```

Step 2 – Investigate the set of sequenced fractional arcs leading to *task* nodes, in ascending order of the time slice associated with the fractional decision variable. Find the first destination that may be feasibly reached.

```
$t^{\text{min}} = \max(T) + 1$ (Initialize to a late time).
for all $(r, i_r, j, t) \in \{Seq : j \in M\}$ do
  % Search the sequenced fractional arcs in ascending order of $t$.
  if $t < t^{\text{min}}$ then
    $(t^{\text{early}}, t'') = \texttt{findEarlyStart}(r, i_r, t_r, j)$
    if $(t^{\text{early}} \neq \varnothing)$ then
      $r' \leftarrow r,\ j' \leftarrow j,\ t' \leftarrow t^{\text{early}},\ t^{\text{min}} \leftarrow t$
    end if
  end if
end for
if no feasible assignment was found then go to Step 3. Else, stop, returning the values for
```

r', j', and t'.

Step 3 – Investigate the set of sequenced fractional arcs leading to *sink* nodes. Select the time-feasible sink destination with the largest corresponding fractional decision variable value.

```
$x^{\max} = 0$ (Initialize to a small value).
for all $(r, i_r, j, t) \in \{Seq : j \notin M\}$ do
  % Search sequenced fractional arcs in descending order of fractional value of $x^t_{r,i_r,j}$:
  if $x^t_{r,i_r,j} > x^{\max}$ then
    $(t^{\text{early}}, t'') = \texttt{findEarlyStart}(r, i_r, t_r, j)$
    if $(t^{\text{early}} \neq \varnothing)$ then
      $r' \leftarrow r$, $j' \leftarrow j$, $t' \leftarrow t^{\text{early}}$, $x^{\max} \leftarrow x^t_{r,i_r,j}$
    end if
  end if
end for
if no feasible assignment was found then go to Step 4. Else, stop, returning the values for
$r'$, $j'$, and $t'$.
```

Step 4 – None of the fractional arcs represent time-feasible assignments for any resource. Select the sequenced fractional arc, leading to a task, with the largest decision variable value.

```
$x^{\max} = 0$ (Initialize to a small value).
for all $(r, i_r, j, t) \in \{Seq : j \in M\}$ do
  % Search sequenced fractional arcs in descending order of fractional value of $x^t_{r,i_r,j}$:
  if $x^t_{r,i_r,j} > x^{\max}$ then
    $r' \leftarrow r$, $j' \leftarrow j$, $t' \leftarrow t^{\text{early}}$, $x^{\max} \leftarrow x^t_{r,i_r,j}$
  end if
end for
```

4.4.2 Rule C1

Due to the penalty assessed on ‘infinite’ resources and ‘infinite’ expendable payload, fractional solutions from the LP relaxation attempt to minimize the values attributed to these decision variables. As a result, the objective function value obtained by the LP relaxation may provide a poor bound. In an effort to obtain a more realistic upper bound, Rule C1 applies a penalty to the objective function value of the LP relaxation. This bound is not necessarily valid, as it is possible that an integer solution that utilizes an IR for a less-penalized task may exist. However, it will always be true that an integer solution constructed from an LP relaxation that contains a non-zero value for an IR will have a non-zero integer value for the IR. Furthermore, if all required tasks have identical priority values (p_j), then the penalty described below will be valid. If Z^{LP} represents the objective function value of the LP relaxation, we apply the penalty as follows:

$$\begin{aligned} Z^{\text{LP}'} &= Z^{\text{LP}} - \sum_{j \in \{RNA \cup RNP\}} p_j \gamma(\lceil y_j \rceil - y_j) \\ &\quad - \sum_{j \in \{RNS \cup RNM(s)\}} \sum_{t \in T_j} p_j \gamma(\lceil z^t_j \rceil - z^t_j) - \sum_{j \in P} p_j \gamma(\lceil w_j \rceil - w_j), \end{aligned}$$

where p_j is the priority of task j, γ represents the scaling parameter used in the objective function term for 'infinite' resources and payload, y_j and z_j^t are the integer decision variables for IR, w_j is the integer decision variable for 'infinite' payload, P is the set of tasks requiring payload delivery, and $\lceil \cdot \rceil$ is the ceiling operator. If the resulting value of $Z^{\mathrm{LP}'}$ is less than the incumbent objective function value, we fathom at the previous node of the branch-and-bound tree.

Aside from the inclusion of the objective function penalty, Rule C1 features additional logic not found in Rule A1. For example, preference is given to pop-up tasks that may be assigned without requiring an early-arriving asset to loiter while waiting for a task's earliest allowable starting service time. Next, preference is given to arcs that were assigned in the initial mission plan. Finally, if no feasible assignment can be found from the *OoS* and *Seq* candidate arcs, the current node of the branch-and-bound tree is preemptively fathomed. Rule C1 may be expressed as follows:

Step 0 – Bound Revision and Initialization:

```
if $Z^{\mathrm{LP}'}$ is less than the incumbent then
    Go to Step 8.
else
    Set $r' = j' = t' = \varnothing$ (No assignment has been determined).
    Set $t^{\mathrm{asgn}} = \max(T) + 1$ (Initialize to a late time).
end if
```

Step 1 – From the set of out-of-sequence fractional arcs, select the earliest pop-up task that may be performed without loitering.

```
for all $(r, i_r, j, t) \in (OoS : j$ is a pop-up task$)$ do
    $(t^{\mathrm{early}}, t'') =$ findEarlyStart$(r, i_r, t_r, j)$
    if $((t^{\mathrm{early}} < t^{\mathrm{asgn}})$ and $(t^{\mathrm{early}} = t_r + f_{r,i_r,j}))$ then
        $r' \leftarrow r,\ j' \leftarrow j,\ t' \leftarrow t,\ t^{\mathrm{asgn}} \leftarrow t''$
    end if
end for
if no feasible assignment was found then go to Step 2. Else, stop, returning the values for
$r'$, $j'$, and $t'$.
```

Step 2 – From the set of out-of-sequence fractional arcs, select the earliest destination that matches an initial assignment.

```
for all $(r, i_r, j, t) \in (OoS : a_{r,i_r,j} = 1)$ do
    $(t^{\mathrm{early}}, t'') =$ findEarlyStart$(r, i_r, t_r, j)$
    if $(t^{\mathrm{early}} < t^{\mathrm{asgn}})$ then
        $r' \leftarrow r,\ j' \leftarrow j,\ t' \leftarrow t,\ t^{\mathrm{asgn}} \leftarrow t''$
    end if
end for
if no feasible assignment was found then go to Step 3. Else, stop, returning the values for
$r'$, $j'$, and $t'$.
```

Step 3 – From the set of out-of-sequence fractional arcs, select the destination node that may be performed earliest.

```
for all $(r, i_r, j, t) \in OoS$ do
    $(t^{\mathrm{early}}, t'') =$ findEarlyStart$(r, i_r, t_r, j)$
    if $(t^{\mathrm{early}} < t^{\mathrm{asgn}})$ then
```

```
      $r' \leftarrow r$, $j' \leftarrow j$, $t' \leftarrow t$, $t^{\text{asgn}} \leftarrow t''$
    end if
  end for
  if no feasible assignment was found then go to Step 4. Else, stop, returning the values for
  $r'$, $j'$, and $t'$.
```

Step 4 – From the set of sequenced fractional arcs, select the earliest pop-up task that may be performed without loitering.

```
  for all $(r, i_r, j, t) \in (Seq : j$ is a pop-up task$)$ do
    $(t^{\text{early}}, t'') = \texttt{findEarlyStart}(r, i_r, t_r, j)$
    if $((t^{\text{early}} < t^{\text{asgn}})$ and $(t^{\text{early}} = t_r + f_{r,i_r,j}))$ then
      $r' \leftarrow r$, $j' \leftarrow j$, $t' \leftarrow t$, $t^{\text{asgn}} \leftarrow t''$
    end if
  end for
  if no feasible assignment was found then go to Step 5. Else, stop, returning the values for
  $r'$, $j'$, and $t'$.
```

Step 5 – From the set of sequenced fractional arcs, select the earliest destination that matches an initial assignment and has a corresponding fractional decision variable value of at least 0.25.

```
  for all $(r', i_{r'}, j', t') \in (Seq : a_{r',i_{r'},j'} = 1$ and $x^{t'}_{r',i_{r'},j'} \geq 0.25)$ do
    $(t^{\text{early}}, t'') = \texttt{findEarlyStart}(r, i_r, t_r, j)$
    if $(t^{\text{early}} < t^{\text{asgn}})$ then
      $r' \leftarrow r$, $j' \leftarrow j$, $t' \leftarrow t$, $t^{\text{asgn}} \leftarrow t''$
    end if
  end for
  if no feasible assignment was found then go to Step 6. Else, stop, returning the values for
  $r'$, $j'$, and $t'$.
```

Step 6 – From the set of sequenced fractional arcs, select the pop-up task that may be performed earliest.

```
  for all $(r, i_r, j, t) \in (Seq : j$ is a pop-up task$)$ do
    $(t^{\text{early}}, t'') = \texttt{findEarlyStart}(r, i_r, t_r, j)$
    if $(t^{\text{early}} < t^{\text{asgn}})$ then
      $r' \leftarrow r$, $j' \leftarrow j$, $t' \leftarrow t$, $t^{\text{asgn}} \leftarrow t''$
    end if
  end for
  if no feasible assignment was found then go to Step 7. Else, stop, returning the values for
  $r'$, $j'$, and $t'$.
```

Step 7 – From the set of sequenced fractional arcs, select the destination that may begin service earliest.

```
  for all $(r, i_r, j, t) \in Seq$ do
    $(t^{\text{early}}, t'') = \texttt{findEarlyStart}(r, i_r, t_r, j)$
    if $(t^{\text{early}} < t^{\text{asgn}})$ then
      $r' \leftarrow r$, $j' \leftarrow j$, $t' \leftarrow t$, $t^{\text{asgn}} \leftarrow t''$
    end if
  end for
```

if no feasible assignment was found **then** go to Step 8. **Else**, stop, returning the values for r', j', and t'.

Step 8 – Artificially (heuristically) fathom:
No time-feasible destinations are available from the fractional arcs.
Fathom the *current* node and take the 'right' branch of the *parent* node.

4.5 Applying the Assignment & Valid Cuts – 'Left' Branch

Both of the proposed branching rules determine values for r', $i_{r'}$, j', and t', assuming that a feasible assignment can be made. These values indicate the resource that is to be assigned along arc $(i_{r'}, j')$, and the earliest feasible starting time for which node j' may be assigned. If no feasible assignment is found, the 'right' branch should be followed.

Using the values determined by a branching rule, the assignment should be implemented in the branching process. As previously mentioned, the nature of the assignment is dictated by the type of node j'. If $j' \in \{RNA \cup ONA \cup B\}$, the assignment is simply to force r' along arc $(i_{r'}, j')$ at time t'. Likewise, if $j' \in RNP$, r' is assigned to task j' at time t'. Because RNP tasks rely upon additional binary decision variables, the values of these variables may also be assigned. Specifically, $x_{t',j'}^{RNP}$ is the binary decision variable that indicates whether task j' is started at time t', and $x_{j'}^{RNP'}$ is the binary decision variable that indicates whether at least one task from the set $RNP'(j')$ is performed.

However, if j' is a time-coordinated task (RNS or $RNM(s)$), two types of assignments are possible. The first type assigns resource r' to arc $(i_{r'}, j')$ at *any* time on or after t'. It also prohibits any other resource assigned to task j' from performing the task before time t'. In this case, the assignment allows the LP relaxation to determine feasible starting times for all resources assigned to task j'. Of course this does not guarantee that an integer assignment of resources to j' at a particular time will be found by the LP relaxation. Therefore, if task j' were selected by a branching rule, and if all candidate resources have been forced to perform task j' (via a previous assignment that routed resources to task j' without specifying a time for service to begin), then the second type of assignment will force these resources to start task j' at a *particular* time t'.

Note that, for this second case, task $j' \in \{RNS \cup RNM(s)\}$ would only be selected by any of the branching rules if all other resources assigned to *other* nodes have assignment times that are *after* $\max(T_{j'})$. Thus, the decision to fix the starting time of task j' is delayed until all other resources would have had the opportunity to be assigned to task j'.

The logic governing the assignment process is as follows:

```
if $j' \in \{RNA \cup ONA \cup B\}$ then
    $x_{r',i_{r'},j'}^{t'} = 1$    [Add assignment to force $r'$ along arc $(i_{r'}, j')$ at time $t'$.]
else if $j' \in RNS$ then
    if $\sum_{t \in T_{j'}: t \geq t'} x_{r',i_{r'},j'}^{t} < 1$ then
        $\sum_{t \in T_{j'}: t \geq t'} x_{r',i_{r'},j'}^{t} = 1$    [Force $r'$ along arc $(i_{r'}, j')$ at some time not prior to $t'$.]
        $h_{j'}^{t} \geq x_{r',i_{r'},j'}^{t} \quad \forall\, t \in \{T_{j'} : t \geq t'\}$    [Ensure $h_{j'}^{t}$ assumes the proper value.]
```

```
      $\sum_{t \in \{T_{j'} : t < t'\}} h_{j'}^{t} = 0$
    else
      $h_{j'}^{t'} = 1$    [Force all resources performing task $j'$ to begin at time $t'$.]
    end if
  else if $j' \in RNM(s)$ for some $s \in S$ then
    if $\sum_{t \in T_{j'} : t \geq t'} x_{r', i_{r'}, j'}^{t} < 1$ then
      $\sum_{t \in T_{j'} : t \geq t'} x_{r', i_{r'}, j'}^{t} = 1$    [Force $r'$ along arc $(i_{r'}, j')$ at some time not prior to $t'$.]
      $b_s^t \geq x_{r', i_{r'}, j'}^{t} \quad \forall t \in \{T_{j'} : t \geq t'\}$    [Ensure $b_s^t$ assumes the proper value.]
      $\sum_{t \in \{T_{j'} : t < t'\}} b_s^t = 0$
    else
      $b_s^{t'} = 1$    [Force all resources performing tasks in set $s \in S$ to begin at time $t'$.]
    end if
  else if $j' \in RNP$ then
    $x_{r', i_{r'}, j'}^{t'} = 1$    [Force $r'$ along arc $(i_{r'}, j')$ at time $t'$.]
    $x_{t', j'}^{RNP} = 1$    [Add constraint to indicate that $RNP$ task $j'$ begins service at time $t'$.]
    if $y_{j'} > 0$ then
      $x_{j'}^{RNP'} = 1$    [Ensure that at least one predecessor task from $RNP'(j')$ is performed.]
    end if
  end if
  if $j' \in RNP'(k)$ for any $k \in RNP$ then
    $x_{j'}^{RNP'} = 1$    [Ensure that successor task $j'$ will be performed if the predecessor is assigned.]
  end if
```

It should be noted that these assignments are determined by the type of task j'. Therefore, problem instances not involving certain task types do not require the use of all conditional (if) statements. Thus, a simplified version of the assignment process may be obtained.

In addition to applying the given assignment, the branching procedure also includes valid cuts to strengthen subsequent solutions to the LP relaxation. These cuts exploit the fact that the resource routes are constructed forward, starting from each resource's source node. Given that resource r' is assigned to node j' above, and that t' represents the earliest feasible time for which service at node j' may begin, the first cut prevents the *next* node following j' from being assigned too early:

$$\sum_{k \in \{\Delta_r^+ : j' \in \Delta_{r',k}^-\}} \sum_{t \in \{T_k : t < t' + f_{r',j',k}\}} x_{r',j',k}^t = 0.$$

Here, resource r' may not be assigned to any arc (j', k) at a time that does not satisfy the travel time requirements between nodes j' and k.

Next, cuts should be added to eliminate other time-infeasible fractional assignments that appeared in the solution to the LP relaxation. Let $A_{r'}$ be defined to be the set of arcs that have non-zero fractional values for resource r' and have times that are prior to t', such that $k \neq i_{r'}$ and $l \neq j'$ for any $(k,l) \in A_{r'}$. For example, suppose the solution to the LP relaxation contains $0 < x_{r',k,l}^t < 1$ for some $(k,l) \in A_{r'}$ and $t < t'$. Based on the assignment above, a route for r' has been constructed through node j' at time t'. Therefore, it would be infeasible for r' to be assigned to any arc in $A_{r'}$ prior to the time required for r' to travel along arcs $(i_{r'}, j')$, (j', k), and (k, l). The valid cuts that prohibit r' from being assigned to arcs in $A_{r'}$ at infeasible times are given by:

$$\sum_{t \in \{T_l : t < t' + f_{r',j',k} + f_{r',k,l}\}} x_{r',k,l}^t = 0 \quad \forall\ (k,l) \in A_{r'},$$

Recall that the $f_{r',j',k}$ parameter captures the travel time for r' along arc (j', k), as well as the service duration at node j', $d_{j'}$. The $f_{r',k,l}$ parameter, then, also captures d_k and the travel time along (k, l).

4.6 Prohibiting the Assignment – 'Right' Branch

When taking a 'right' branch in the branch-and-bound tree, the single constraint is given by:

$$\sum_{t \in T_j} x_{r',i_{r'},j'}^t = 0,$$

for the given r', $i_{r'}$, and j'. Thus, resource r' is prohibited from taking arc $(i_{r'}, j')$ at *any* time.

We note that [8] employ a similar approach of either requiring or prohibiting an assignment by branching on a summation of decision variables, in the context of a manpower allocation problem with job-teaming constraints. However, due to the nature of their problem formulation, their approach also requires branching on a continuous decision variable representing the assignment time of team members to accomplish team coordination.

5 Solution Approach

Preliminary testing and tuning of both branching rules, *RadScale* values, and α values indicated that neither rule was clearly dominant across all problem instances. This testing also showed the state space reduction approach to be a beneficial aspect of our solution procedure. Furthermore, we found that the speed and quality of solutions are influenced by the value of the α parameter, which dictates the relative degree of trade-off between maximizing overall mission effectiveness ($\alpha = 1$) and minimizing changes to the initial mission plan ($\alpha = 0$). The probable explanation for this behavior is that smaller α values correspond to a higher reliance on the initial mission plan, whereby the initial plan tends to 'guide' the LP relaxation toward integer valued solutions. In Section 6, results are reported for instances of $\alpha = 0.25$ and $\alpha = 0.75$.

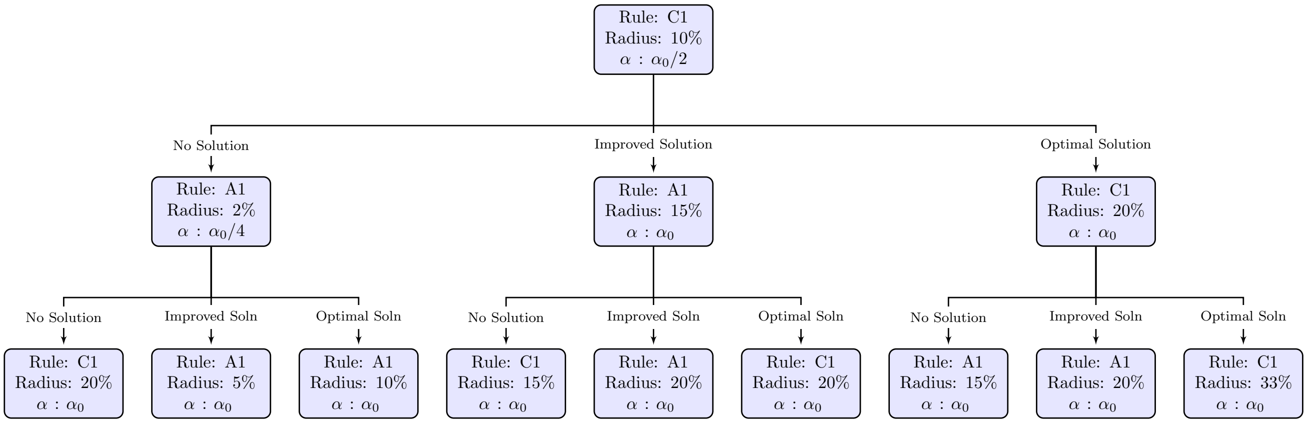


Figure 3: Algorithm Flowchart

Although neither branching rule promises to be most effective for all problems, the initial testing revealed that at least one rule produces improved solutions for any given problem. Therefore, our algorithm attempts to solve the dynamic airborne resource rerouting problem in three phases, such that different combinations of branching rules and solution spaces may be employed. The maximum allowable runtime for the algorithm, denoted as $maxTime$, is pre-specified by the user. As such, each phase has a runtime limit of $maxTime/3$. Each phase is characterized by the selection of a branching rule, a radius scale for solution space reduction, and an augmentation of the α parameter.

Because Rule C1 had marginally superior performance on the preliminary test problems, both in terms of solution quality and runtime, it was chosen to be the branching rule applied in the first phase. Similarly, a $RadScale$ value of 10% was selected for the first phase because it was the largest value that did not result in a noticeable decline in solution quality. Finally, given that smaller α values tend to produce better solutions, the objective function in the first phase replaces α by $\alpha_0/2$, where α_0 represents the decision maker's preferred (pre-specified) value of α. Upon selecting these algorithm parameters, the 'intelligent' branch-and-bound procedure commences, where the starting incumbent solution is determined by the initial mission plan. The branch-and-bound procedure is executed until either an optimal solution is found, or the runtime reaches a limit of $maxTime/3$.

The algorithm parameters employed by phase two of the algorithm are determined by the quality of the solution obtained in phase one. If no improved solution was found, it may be the result of Rule C1 being ineffective for this problem, the solution space being too expansive, or the α value being too large. Due to the time critical nature of the dynamic rerouting problem, it is impractical to change these parameters individually. Instead, all three parameters are changed such that Rule A1 is employed, $RadScale$ is reduced to 2%, and α is lowered to $\alpha/4$. If an improved, albeit not provably optimal, solution was found in phase one, we again change the algorithm parameters in phase two. However, this time we increase both the size of the solution space (to consider additional candidate arcs) and the value of α (to match the original input value). We also change the branching rule to Rule A1. Finally, if the optimality condition is reached in phase one, we continue to apply Rule C1 in phase two. However, the value of $RadScale$ is increased to explore a larger solution space. Furthermore, α is reset to its original input value to be consistent with the decision-maker's preference for the relative degree to which the objectives of maximizing overall mission effectiveness and minimizing changes to the initial mission plan are weighted.

A similar process is applied in the third phase of the algorithm, where the algorithm parameters are determined by the performance of the algorithm in phase two. Figure 3 shows a flowchart of algorithm parameter settings, where each row of the flowchart corresponds to one of the three phases. Based on this flowchart, the proposed algorithm may be outlined as follows:

Initialization Specify the maximum allowable runtime for the algorithm, $maxTime$.

Phase 1 Select Rule C1 for branching, apply the solution space reduction technique using $RadScale = 0.10$, and set $\alpha = \alpha_0/2$.

- Use the initial mission plan to determine the incumbent solution.
- Execute the branch-and-bound procedure until either
  - the optimal solution for this phase is found, or
  - the runtime reaches $maxTime/3$.

Phase 2 Based on the solution obtained in the previous phase, select the appropriate branching rule, apply the solution space reduction procedure according to $RadScale$, and update the value of α.

- Use the best known solution from the previous phase to determine the incumbent solution, noting that if the value of α has changed, so to will the value of the objective function.
- Execute the branch-and-bound procedure until either
  - the optimal solution for this phase is found, or
  - the runtime reaches $maxTime/3$.

Phase 3 Repeat Phase 2, selecting the branching rule, $RadScale$, and α values from the appropriate block in the bottom row of Figure 3.

6 Numerical Analysis

6.1 Problem Generation

Because there are no applicable benchmark problems on which to test our solution approach, test problems were created. Readers interested in how these test problems were generated may refer to [16] or contact the authors directly.

Table 4 shows the major parameter settings that were adjusted to create our test problems. Each combination of parameter values is used to generate three replications. For example, for problems involving 15 initial tasks, 72 distinct problems were generated by the combinations of parameter settings. Initially-assigned tasks have not been completed as of the time that the pop-up event occurred. The location of each task is generated uniformly across a battlespace area of size ($\sim$unif(50,400)) × ($\sim$ unif(50,400)). The resulting battlespace sizes range from approximately the area of Oklahoma City to the area of the state of California. Two base locations are defined for each problem. The 'time window scale' refers to the width of each task's time window, as a percentage of the initial mission horizon. Although our model and solution approach are capable of accommodating disjoint time windows, in this analysis each task has a single time window composed of consecutive time intervals. The Loiter Time is the amount of slack that is built into the initial mission plan at each task. It is uniformly generated over the range of values. For problems involving a "none" payload value, no tasks require payload delivery. For problems involving "low" payload, the minimum payload quantity to be delivered, $k_j^{\min}$, was determined according to Table 5.

Each task j is characterized by several parameters. The actual value of each parameter may be drawn from a distribution, as shown in Table 5. The task duration, d_j, is measured in units of

Table 4: Design of Experiments

Parameter	**Values**		
Initial Tasks	15	30	50
Resources	2	2, 4	3, 5
Pop-ups	2, 6		
Time Window Scale	5%, 10%, 25%		
Loiter Time [minutes]	∼unif(0,10), ∼ unif(5,30)		
Payload	None, Low		

Table 5: Task Parameters

Parameter	**Value**	**Probability**
d_j	0	0.50
	∼ unif(1,5)	0.20
	∼ unif(5,10)	0.20
	∼ unif(10,20)	0.10
p_j	10 (high)	0.35
	5 (medium)	0.55
	1 (low)	0.10
$n_j^{\max}$	1	0.70
	2	0.30
$k_j^{\min}$	0	0.85
	1	0.10
	2	0.05

minutes. The task priority, p_j, is used to determine whether a task is required or optional, such that all low-priority tasks are assumed to be optional (i.e., of the ONA task type), while medium- and high-priority tasks are assumed to be required. The maximum number of resources that may be assigned to task j is given by $n_j^{\max}$. For required tasks, the minimum number of resources, $n_j^{\min}$, is $\sim$ unif$(1, n_j^{\max})$, while $n_j^{\min} = 0$ for optional tasks. Finally, the minimum required payload delivery is represented by $k_j^{\min}$.

The current location of each resource at the time of the pop-up event, represented by node Δ_r^0, is generated uniformly across the battlespace area to simulate the event that the resources are already in flight when pop-ups are identified. Resources are classified as being one of three 'types,' according to the following probabilities: Resources are of type 1 with probability 0.50, of type 2 with probability 0.40, and of type 3 with probability 0.10. Resource type 1 has a velocity of 170 miles per hour, while types 2 and 3 have velocities of 150 and 130, respectively. Using these resource types, effectiveness values for a given task may be assigned as per Table 6.

Table 6: Task Effectiveness by Resource Type

Task Effectiveness	**Probability**
5 (highly effective)	0.15
3 (moderately effective)	0.30
1 (barely effective)	0.30
N/A (unable to perform this task)	0.25

Table 7: Summary of algorithm performance.

Initial	Solution	Time [seconds]			OFV		% Impr		
Tasks	Approach	Mean	Std Dev	Limit	Mean	Std Dev	Mean	Std Dev	% Failed to Improve
15	Heuristic	13.57	13.26	30	-57.09	87.43	86.97	20.4	0.00 (0 of 144)
	CPLEX	13.39	12.49	30	-82.86	131.11	81.53	28.44	7.64 (11 of 144)
30	Heuristic	24.65	9.78	30	-145.38	164.83	63.32	35.55	7.99 (23 of 288)
	CPLEX	23.65	10.28	30	-234.27	264.43	45.31	43.41	42.36 (122 of 288)
50	Heuristic	26.93	7.43	30	-235.69	251.83	51.78	37.94	13.54 (39 of 288)
	CPLEX	27.59	6.30	30	-368.60	267.61	17.49	32.89	74.31 (214 of 288)

6.2 Algorithm Performance

In this section the proposed algorithm is compared to CPLEX's MIP solver. Each test problem utilizes a time interval size of 1% of the mission horizon. For example, a 5-hour mission would be partitioned into 3-minute time intervals, offering a high degree of granularity. An IR penalty of 10, matching the value of the highest-priority task, is applied. The β parameter, which governs the degree to which the objective function is weighted regarding the minimization of total travel time, is set to 0.7. The value of α is tested at values of 0.25 and 0.75. The maximum runtime for our algorithm and CPLEX is 30 seconds for each problem.

CPLEX's callable library functions are utilized to add and remove constraints in our branch-and-bound algorithm. This relies on the LP relaxation of the problem, and our constraints are added to elicit an integer-valued solution. The branch tolerance for our algorithm is set to 0.5%, while CPLEX's default tolerance is 0. Although this appears to put our algorithm at an advantage, initial testing revealed that CPLEX demonstrated better performance with a tolerance of 0, rather than 0.5% (recall that solution time is limited in any case). All test problems were conducted using CPLEX 12.2.0 on an HP 8100 Elite desktop PC, with a quad-core Intel i7-860 processor running Ubuntu Linux 10.10 in 64-bit mode.

Figure 4 provides a summary of the percentage of test problems for which each approach was able to find an improved solution over the initial mission plan. Table 7 provides a more detailed comparison of our algorithm to CPLEX's MIP solver, where 'OFV' represents the objective function value, '% Impr' represents the percentage improvement of the solution over the initial mission plan (our first incumbent solution), and '% Failed to Improve' represents the percentage of problems for which the solution approach did not find any solutions that were better than the initial mission plan. It is observed that, as the problem size increases, this percentage grows dramatically for CPLEX. These results indicate that our algorithm outperforms CPLEX in terms of solution quality across all problem sizes. However, in terms of solution time, the two approaches are very similar. This is not surprising, given the limited runtime allowance for these difficult problems.

Figure 5 contains summary boxplots, grouped by the number of initially-assigned tasks in each data set.

7 Summary and Future Research

An algorithm for dynamically rerouting and rescheduling airborne platforms was presented. This is an improved branch-and-bound algorithm in which additional cuts are added to tighten subsequent bounds. In this algorithm, resource routes are constructed from source nodes to sink nodes, where resources are assigned to nodes at the earliest feasible time. The assignments made in each node of the branch-and-bound tree are based on the task type.

The proposed algorithm employs a solution space reduction procedure that reflects the spa-

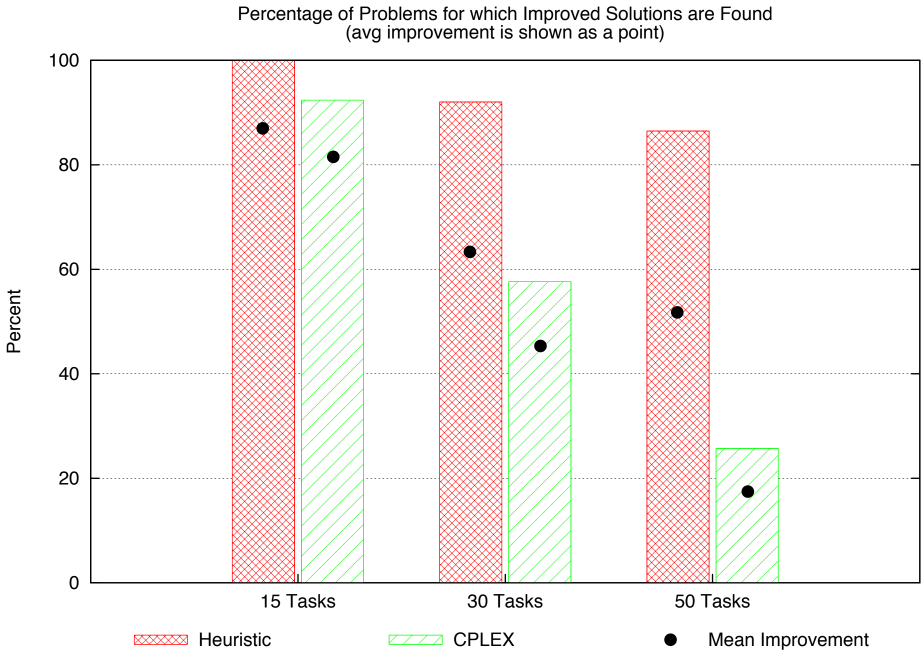
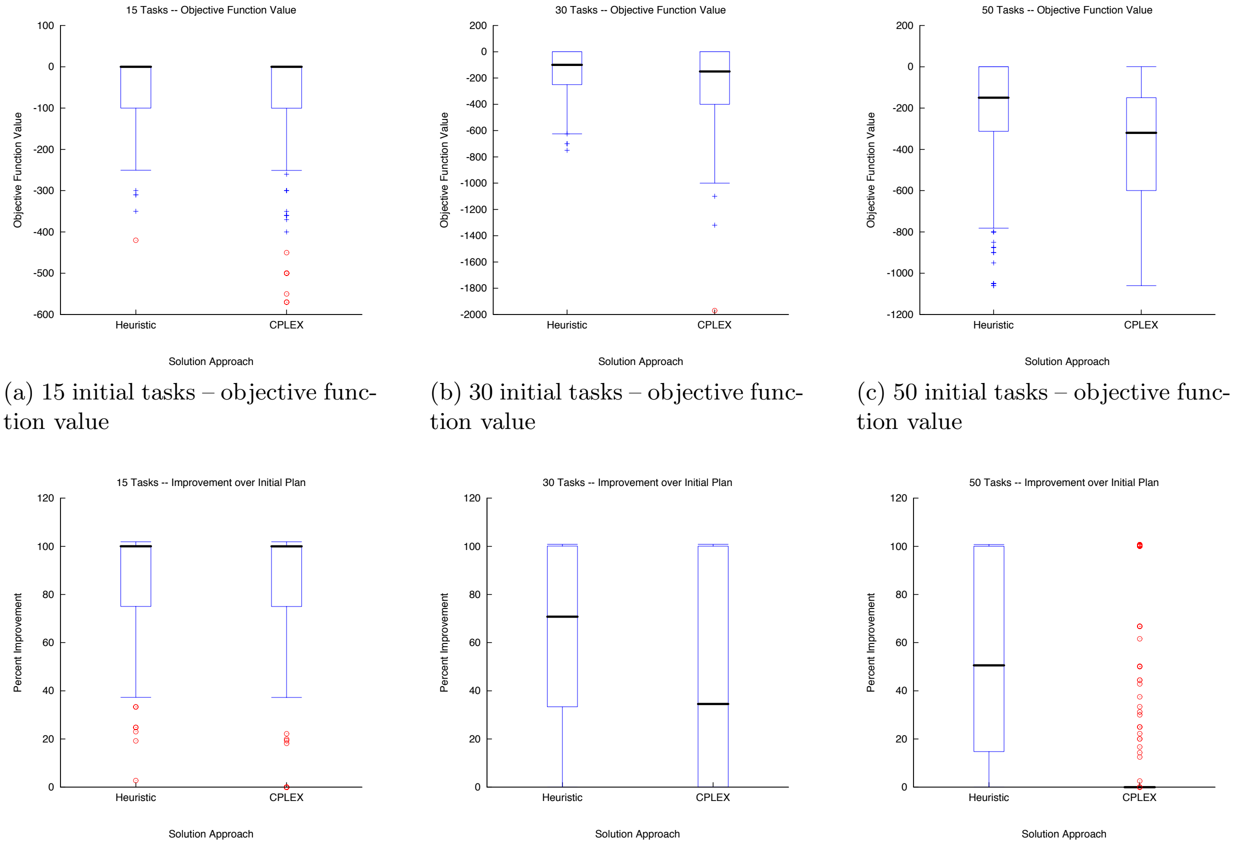


Figure 4: Summary of algorithm performance.

tiotemporal aspects of the initial mission plan and two different branching rules. One of the key features of the branching rules is that decision variable assignments that occur during the branching process do not require the selection of a decision variable containing a non-zero fractional value in the LP relaxation. Instead, the branching rules use the LP relaxation to find candidate *arcs* between nodes in the network, and then the appropriate resource is routed to the next node at the earliest feasible time. As a result, the effective solution space is reduced. Additionally, rather than setting a single decision variable value equal to zero when taking the 'right' (prohibit) branch in the branch-and-bound tree, the approach prohibits a resource from taking an arc at any time. Numerical analysis indicates that this approach outperforms CPLEX on a set of realistically-sized problems and provides improved solutions in a limited time.

Although the performance results are encouraging for large-scale problems of up to 50 tasks, myriad opportunities for further study exist. First, the authors plan to evaluate the proposed solution approach on a variety of other vehicle routing problems that do not require complex coordinated tasks. Specifically, this may include the *dynamic vehicle routing problem* (c.f., [10]) and the *dynamic assignment problem* (c.f., [23]). This should require only minor modifications beyond simply removing the elements of the algorithm that explicitly account for the unused task types. Such research may also afford the opportunity to integrate additional branching rules into the algorithm. Previous testing (see [16]) indicates that combinations of additional branching rules may offer even greater performance, although at the expense of increased algorithmic complexity. Additionally, modified solution space reduction techniques could prove to be beneficial. Furthermore, as this work represents the first solution approach for this dynamic rerouting problem involving complex task types, there are opportunities to develop alternative solution techniques, including metaheuristics, and compare these against the proposed algorithm. For still-larger problems, the branch-and-bound approach may prove to be time prohibitive. Although there is always the possibility of employing faster computing power, it is likely the case that new solution approaches will



(a) 15 initial tasks – objective function value

(b) 30 initial tasks – objective function value

(c) 50 initial tasks – objective function value

(d) 15 initial tasks – percentage improvement of objective function

(e) 30 initial tasks – percentage improvement of objective function

(f) 50 initial tasks – percentage improvement of objective function

Figure 5: Comparison of solution quality for test problems involving 15, 30, and 50 initial tasks.

be required for very large-scale problems.

Acknowledgements

This research has been funded partially by Office of Naval Research under contract number N00173-08-C-4009, Air Force Research Laboratories through Sierra Nevada Corporation, and Air Force Research Laboratories through IAVO Corp. The authors wish to thank the two anonymous reviewers and the Associate Editor, whose helpful suggestions contributed to an improved manuscript.

References

[1] J. Bellingham, M. Tillerson, A. Richards, and J. P. How. *Cooperative Control: Models, Applications and Algorithms*, chapter 2 – Multi-Task Allocation and Path Planning for Cooperating UAVs, pages 23–39. Kluwer Academic Publishers, Boston, 2003.

[2] B. Bennett. Police employ predator drone spy planes on home front. *Los Angeles Times*, December 10, 2011. URL `http://articles.latimes.com/2011/dec/10/nation/la-na-drone-arrest-20111211`.

[3] J. Berger, M. Barkaoui, and A. Boukhtouta. A hybrid genetic approach for airborne sensor vehicle routing in real-time reconnaissance missions. *Aerospace Science and Technology*, 11:317–326, 2007.

[4] F. Bullo, E. Frazzoli, M. Pavone, K. Savla, and S.L. Smith. Dynamic vehicle routing for robotic systems. *Proceedings of the IEEE*, 99(9):1482–1504, September 2011.

[5] D.W. Casbeer and R.W. Holsapple. Column generation for a UAV assignment problem with precedence constraints. *International Journal of Robust and Nonlinear Control*, 21(12):1421–1433, 2011.

[6] G. Clarke and JW Wright. Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, 12:568–581, 1964.

[7] J-F Cordeau, G. Desaulniers, J. Desrosiers, M. Solomon, and F. Soumis. *VRP with Time Windows*, chapter 7, pages 157–194. SIAM, 2002.

[8] A. Dohn, E. Kolind, and J. Clausen. The manpower allocation problem with time windows and job-teaming constraints: A branch-and-price approach. *Computers & Operations Research*, 36(4):1145–1157, 2009.

[9] M. Drexl. Synchronization in vehicle routing – a survey of VRPs with multiple synchronization constraints. *Transportation Science*, published online before print March 8, 2012, 2012.

[10] T. Flatberg, G. Hasle, O. Kloster, E.J. Nilssen, and A. Riise. Dynamic and stochastic aspects in vehicle routing-a literature survey. Technical report, SINTEF Technical Report STF90A05413, 2005.

[11] J. Jackson, A. Girard, S. Rasmussen, and C. Schumacher. A combined tabu search and 2-opt heuristic for multiple vehicle routing. In *American Control Conference (ACC)*, pages 3842–3847, July 2010.

[12] F. Janez. Optimization method for sensor planning. *Aerospace Science and Technology*, 11:310–316, 2007.

[13] S. Karaman and E. Frazzoli. Vehicle routing problem with metric temporal logic specifications. In *Decision and Control, 2008. CDC 2008. 47th IEEE Conference on*, pages 3953–3958. IEEE, 2008.

[14] S. Karaman and E. Frazzoli. Linear temporal logic vehicle routing with applications to multi-UAV mission planning. *International Journal of Robust and Nonlinear Control*, 21(12):1372–1395, 2011.

[15] G.W. Kinney, R.R. Hill, and J.T. Moore. Devising a quick-running heuristic for an unmanned aerial vehicle (UAV) routing system. *Journal of the Operational Research Society*, 56:776–786, 2005.

[16] C.C. Murray. *Dynamic Reassignment and Rerouting in Cooperative Airborne Operations*. PhD thesis, University at Buffalo, The State University of New York, April 2010.

[17] C.C. Murray and M.H. Karwan. An extensible modeling framework for dynamic reassignment and rerouting in cooperative airborne operations. *Naval Research Logistics*, 57(7):634–652, 2010.

[18] C. Schumacher, P.R. Chandler, M. Pachter, and L.S. Pachter. Optimization of air vehicles operations using mixed-integer linear programming. *Journal of the Operational Research Society*, 58(4):516–527, 2007.

[19] A. Shalal-Esa. Haiti imagery spurring interest in Northrop drone. *Reuters*, April 13, 2010. URL http://www.reuters.com/article/2010/04/14/northrop-unmanned-idUSN1318515720100414.

[20] V.K. Shetty, M. Sudit, and R. Nagi. Priority-based assignment and routing of a fleet of unmanned combat aerial vehicles. *Computers & Operations Research*, 35:1813–1828, 2008.

[21] T. Shima, S.J. Rasmussen, A.G. Sparks, and K.M. Passino. Multiple task assignments for cooperating uninhabited aerial vehicles using genetic algorithms. *Computers & Operations Research*, 33:3252–3269, 2006.

[22] M.M. Solomon. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, 35(2):254–265, 1987.

[23] Michael Z. Spivey and Warren B. Powell. The dynamic assignment problem. *Transportation Science*, 38(4):399–419, November 2004.

[24] B. Stamps. Search and rescue goes high tech, April 2, 2012. URL http://app1.kuhf.org/articles/1333401151-Search-And-Rescue-Goes-High-Tech.html.

[25] M. Tavana, M.D. Bailey, and T.E. Busch. A multi-criteria vehicle-target allocation assessment model for network-centric joint air operations. *International Journal of Operational Research*, 3(3):235–254, 2008.

[26] J. Taylor. Drones to patrol the skies above Olympic Stadium. *The Independent*, November 25, 2011. URL http://www.independent.co.uk/news/uk/crime/drones-to-patrol-the-skies-above-olympic-stadium-6267107.html.

[27] A.L. Weinstein and C. Schumacher. UAV scheduling via the vehicle routing problem with time windows. Technical Report AFRL-VA-WP-TP-2007-306, Air Force Research Laboratory, January 2007. Conference paper submitted to the Proceedings of the 2007 AIAA Infotech Aerospace Conference and Exhibit.

[28] A.K. Whitten, H.L. Choi, L.B. Johnson, and J.P. How. Decentralized task allocation with coupled constraints in complex missions. In *American Control Conference*, pages 1642–1649, 2011.

[29] J. Wilde, D. DiBiaso, and M. Nervegna. Team planning for unmanned vehicles in the risk-aware mixed-initiative dynamic replanning system. *Oceans 2007*, pages 1–8, 2007.