



Operational Planning for Multiple Heterogeneous Unmanned Aerial Vehicles in Three Dimensions

Author: Blair Leake Negrón, Detachment 3, 53^d Test Management Group, United States Air Force
Co-Authors: Dr. Stephan Kolitz, Draper Laboratory, Inc and Dr. Hamsa Balakrishnan, Massachusetts Institute of Technology

The incorporation of unmanned aerial vehicles (UAVs) into an increasing variety of applications has resulted in a need for operations planning methods that include the complex constraints of specific applications. The research presented in this paper concentrates on the planning of operations for UAVs for the purpose of monitoring Earth's phenomena through data collection.

The problem presented in this research focuses on developing an operations plan for multiple, heterogeneous UAVs to collect data from spatially distributed locations. The planning of UAV operations for this purpose requires complex constraints to accurately represent the operation. These constraints include: planning in three dimensions to take full advantage of the UAVs' abilities, ensuring data is collected during the time period in which the phenomena occurs, and fully utilizing each UAV's performance capabilities.

I. UAV Operations Planning Problem

This section presents the operations planning problem discussed in this paper; the UAV Operations Planning Problem. The purpose of the UAV Operations Planning Problem is to develop an operations plan to collect observation data that incorporates multiple, heterogeneous UAVs. The planning encompasses three dimensions and time windows for data collection. The goal of the problem is to develop a plan that includes collection of the most important data, as defined by a value associated with each data collection.

Two significant components of problem are *tasks* and *locations*. Tasks are actions to be performed by the UAVs, in the form of data collection. For example, a task could be to obtain an aerial image of a specified area or to collect the air temperature at a given altitude. The concept of a task is broad and dependent on the sensors attached to the UAV. A location is the latitude, longitude, and altitude coordinates at which a task can be performed.

Tasks can be performed at multiple locations where it is possible to collect the observation data. To create a mathematical model, the number of possible locations per task is finite. Increasing the number of possible locations for each task greatly increases the size of the problem. For example, given the task to take an aerial image of a location, the UAV is able to obtain the image from the given coordinates an altitude of 10,000 feet or an altitude of 15,000 feet. The coordinates combined with the given altitudes define two precise locations at which the UAV can obtain the requested data.

To define the importance of the observation, each task also has an assigned value. This value can vary depending on the location that a task is completed. For example, an aerial image can be more valuable based on the altitude from which it is obtained due to resolution and field of view considerations. This value is used to prioritize between the tasks when developing the operations plan.

The operations plan developed for the UAV Operations Planning Problem incorporates multiple UAVs with differing performance specifications. Each UAV in the system is included in the operations plan, although not all UAVs must be utilized at all times during operations.

To model the UAV Operations Planning Problem, the necessary characteristics of the tasks, locations, and UAVs are described mathematically. The components of the UAV Operations Planning

Problem are described by two types of input: (1) input to describe the tasks, and (2) input to describe each UAV. The following input describes each task:

1. *Task Locations*: The set of latitudes, longitudes, and altitudes at which a task could be performed.
2. *Observation Time*: The time it takes for the UAV to complete the task (in hours).
3. *Value*: A measure of the importance of the collected data. The value can be different for different task locations.
4. *Time Windows*: The start and end time during which the observation task has value. For the model, the absolute time windows are converted into the number of hours after start time. For example, an early time window of 1.5 indicates that the task must be performed more than an hour and a half from the start time for the plan (time 0).

The following input describes the performance specifications for each UAV:

1. *Endurance*: The max flight time for each UAV.
2. *Speed*: How fast the UAV moves over ground; constant speed assumed for travel between tasks.
3. *Climb/Sink Rate*: How fast the UAV can increase or decrease its altitude.
4. *Ceiling/Floor*: Operational constraints on the (upper/lower) altitude limits for the UAV.

The solution to the UAV Operations Planning Problem is a plan for the UAVs to collect the data requested for tasks. The tasks' values are used to prioritize which tasks to include in the operations plan. The goal of the UAV Operations Planning Problem is to develop an operations plan that collects the most valuable data. The operations plan is described by the following characteristics:

1. *Task Order*: The order in which the UAV completes its assigned tasks.
2. *Location Selection*: The path plan determines the location at which each task is completed.
3. *Start Times*: The time at which the UAV begins each task assigned to it.
4. *End Times*: The time at which the UAV completes each task assigned to it.
5. *Departure/Return Times*: The time at which the UAV departs its command station and the time at which it returns to its command station.

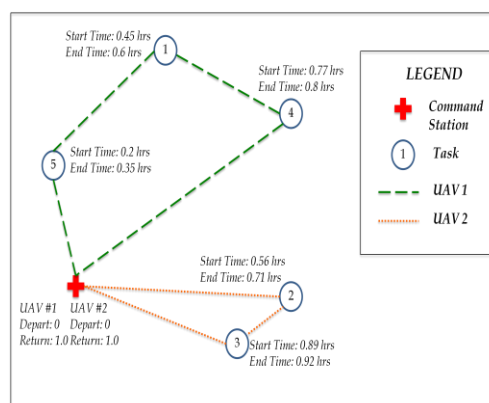


Figure 1. Example Operations Plan.

II. Mixed Integer Programming Formulation

A mixed integer program (MIP) formulation for the UAV Operations Planning Problem provides the framework to evaluate linear programming solution methods. The MIP includes all the constraints necessary to develop a feasible operations plan. The following sets are used in the formulation:

- C = Set of task, location pairs
- T = Set of all tasks
- U = Set of UAVs
- P = Set of placements in a path

The set P is the set of placements in a path. A *placement* denotes the order of a task in the path. For example, if a task is in placement five in a path, then the task is the fifth task that will be performed by the UAV. The set of placements contains the placement for each task in the path; therefore, the set associated with a path with ten tasks will contain the first ten natural numbers while the set of placements associated with a path containing only five tasks will contain the first five natural numbers.

The mixed integer program utilizes four types of decision variables. The decision variable $perform_{(i,k),u,p}$ is modeled after a decision variable in a mixed integer program developed by Miller⁵ to solve an operations planning problem for unmanned surface vehicles (USVs). In Miller's formulation, the variable x_{ikt} takes a value 1 if node i is visited by USV k in the t -th placement on the route. Each decision variable is described in the following list:

- $perform_{(i,k),u,p}$ A binary decision variable describing if task i is performed at location k by UAV u in placement p . The p describes the placement of the task in the respective UAV's path. For example, if $p=1$, then task i at location k will be the first task completed by UAV u .
- $travel_{(i,k),(j,l),u}$ A binary decision variable describing if the arc from (i,k) to (j,l) is traveled by UAV u .
- $arrive_{(i,k),u}$ A continuous variable that assigns the time that UAV u will arrive at task i in location k .
- $depart_{(i,k),u}$ A continuous variable that assigns the time that UAV u will depart task i in location k .

The inputs to the model are described in Section I. They are denoted as:

- $early_i$ Beginning of time window for task i
- $late_i$ End of time window for task i
- $obstime_i$ Required time to complete task i
- $altitude_{(i,k)}$ The altitude of task i at location k
- $horizon$ Planning horizon
- $ceiling_u$ Maximum altitude for UAV u
- $floor_u$ Minimum altitude for UAV u
- $endurance_u$ Maximum length of flight for UAV u
- $traveltime_{(i,k),(j,l),u}$ Length of time for UAV u to travel from location (i,k) to (j,l)
- $value_{(i,k),u}$ The value for UAV u to complete task i at location k

The objective is to maximize the total value of tasks completed. Mathematically, this is represented as:

$$\max \sum_{u \in U} \left(\sum_{(i,k) \in C, p \in P} value_{(i,k),u} * perform_{(i,k),u,p} \right)$$

The mixed integer formulation has thirteen constraints that are categorized as either UAV operational constraints, network constraints, or time window constraints. The UAV constraints ensure the capabilities of the UAVs are not exceeded. The following are the UAV operational constraints:

- (1) To ensure that the observation altitude is above the UAV floor

$$altitude_{(i,k)} * \sum_{u \in U} \sum_{p \in P} perform_{(i,k),u,p} \geq floor_u \quad \forall (i,k) \in C, u \in U$$

- (2) To ensure observation altitude is below UAV ceiling

$$altitude_{(i,k)} * \sum_{u \in U} \sum_{p \in P} perform_{(i,k),u,p} \leq ceiling_u \quad \forall (i,k) \in C, u \in U$$

- (3) To make sure that all activities are completed within the horizon:

$$arrive_{(i,k),u} + obstime_i \leq horizon \quad \forall (i,k) \in C, u \in U$$

The network constraints ensure that the resulting operations plan creates a feasible path plan for the operations plan. The following are the network constraints:

- (4) Each placement, p , can only be assigned one task

$$\sum_{(i,k) \in C} perform_{(i,k),u,p} \leq 1 \quad \forall u \in U, p \in P$$

- (5) Constrain a single UAV to perform a task

$$\sum_{u \in U} \sum_{k \in L} perform_{(i,k),u,p} \leq 1 \quad \forall i \in T, p \in P$$

- (6) Ensure that the targets are assigned in successive placements on the UAV path

$$\sum_{(i,k) \in C} perform_{(i,k),u,p} - \sum_{(i,k) \in C} perform_{(i,k),u,p-1} \leq 1 \quad \forall u \in U, p \in P - 1$$

- (7) Force an arc to exist between two successively performed tasks

$$perform_{(i,k),(u,p+1)} + perform_{(i,k),(u,p)} - travel_{(i,k),(j,l),u} \leq 0 \quad \forall (i,k) \in C, (j,l) \in C, u \in U, p \in P - 1$$

- (8) Force an arc to leave every performed node

$$\sum_{u \in U} travel_{(i,k),(j,l),u} - \sum_{u \in U} \sum_{p \in P} perform_{(i,k),u,p} \leq 0 \quad \forall (i,k) \in C, (j,l) \in C$$

- (9) Make sure than arc enters every performed node

$$\sum_{u \in U} travel_{(i,k),(j,l),u} - \sum_{u \in U} \sum_{p \in P} perform_{(j,l),u,p} \leq 0 \quad \forall (i,k) \in C, (j,l) \in C$$

The time window constraints limit the resulting operations plan to performing tasks within the desired time window. In this formulation, the observation must be completed within the time window. The following are the time window constraints:

(10) UAV must arrive after the beginning of the time window

$$arrive_{(i,k),u} \geq early_i * perform_{(i,k),u,p} \quad \forall (i,k) \in C, u \in U, p \in P$$

(11) UAV must exit before end of time window

$$depart_{(i,k),u} \geq late_i * perform_{(i,k),u,p} \quad \forall (i,k) \in C, u \in U, p \in P$$

(12) UAV must depart the task after sufficient time to complete the task

$$depart_{(i,k),u} \geq arrive_{(i,k),u} + obstime_i * perform_{(i,k),u,p} \quad \forall (i,k) \in C, u \in U, p \in P$$

(13) Ensure sufficient travel time between tasks ($a_{(i,k),(j,l),u}$ must be calculated a priori as explained in the proceeding paragraph)

$$depart_{(i,k),u} + traveltime_{(i,k),(j,l),u} - a_{(i,k),(j,l),u}(1 - travel_{(i,k),(j,l),u}) \leq arrive_{(i,k),u} \quad \forall (i,k) \in C, (j,l) \in C$$

The value for $a_{(i,k),(j,l),u}$ in constraint (13) is a linearization of a necessary constraint and must be calculated before solving the program. The model needs to constrain the time that a UAV arrives at a next task to be greater than the time that it departs the previous task plus the travel time between the two. Ropke, Cordeau, and Laporte⁸, developed the linearization using the following variable:

$$a_{(i,k),(j,l),u} = \max(0, late_i + obstime_i + traveltime_{(i,k),(j,l),u} - early_j)$$

The MIP provides the framework to explore solution methods for the UAV Operations Planning Problem. Exact solution methods can be used to directly solve the MIP; however, due to the complexity of the constraints, which results in long run times, it is beneficial to develop another approach. The next section presents a heuristic algorithm for solving the UAV Operations Planning Problem.

III. Composite Operations Planning Algorithm

The *Composite Operations Planning Algorithm (COPA)* combines the use of a linear program and a metaheuristic to solve the UAV Operations Planning Problem. It combines multiple heuristics with a linear program to solve for a solution to the UAV Operations Planning Problem.

COPA utilizes composite variables, which represent multiple decisions in a single binary variable. The composite variables in COPA represent two parts: a full path and a type of UAV. The full path is an ordered set of tasks that spans the planning horizon. Therefore, a UAV can only perform one full path per planning horizon. The full path composite also has an assigned type of UAV. If the

composite is incorporated into the operations plan, i.e., the composite variable takes a value of 1, then a UAV of the type will be assigned to the path.

COPA includes three main steps:

Step I: Subset Allocation. *In this step, tasks are allocated to be performed by a specific UAV type.*

Step II: Composite Generation. *Using the set of tasks allocated to a UAV type, full path composites are generated for each UAV of the type. This step utilizes a heuristic to create the path plans.*

Step III: Full Path Composite Variable Linear Program. *The full path composites generated in Step II are modeled by full path composite variables in the linear program. The linear program is solved to determine the best set of composites to incorporate into the operations plan for the system of UAVs.*

Steps I and II are iterated until a predetermined number of composite are generated for the linear program; this number might be limited by computer memory or runtime constraints. Step III is solved a single time and the solution is used to an operations plan for the entire set of UAVs.

In the first step of COPA, tasks are assigned to be performed by a specific type of UAV. The result is a mutually exclusive, collectively exhaustive set of task subsets, i.e., a partition of the full task set. Each subset has an assigned type of UAV to perform the tasks in the subset. The objective of this step is to assign tasks to subsets that result in high value composite variable path plans. To achieve this, the assignment of tasks to subsets occurs multiple times and each time a different heuristic method is used to determine to which subset to assign a task. Three heuristics are used: (1) a heuristic based on the observation time necessary to collect the data requested for the task; (2) a heuristic that uses the start time of the task's time window; and (3) randomly assigning tasks to UAV types.

The first heuristic method for subset allocation utilizes the required observation time of the task to determine which type of UAV should perform the task. Tasks with longer observation times are assigned to the subsets associated with the UAVs with longer endurances, while tasks with shorter observation times are assigned to be performed by UAVs with shorter endurances.

The second method for subset allocation utilizes the time windows of each task to determine to which subset to assign the task. The logic behind this method is that each UAV should have tasks whose time windows are spread across the planning horizon. This addresses the problem of UAV types being assigned tasks that are clustered in a portion of the planning horizon, thus unnecessarily limiting the number of tasks that can be completed. The tasks are ordered by the start time of their time window, then sequentially assigning the tasks to UAV types.

This method for creating subsets uses random numbers to assign the tasks to subsets. Depending on the value of the random number the task being assessed is assigned to any of the possible UAV types, if it is feasible for the UAV type to perform the task. The size of the subsets is randomly generated.

The second step of COPA is to generate full path composites from the tasks within the subsets created in Step I. The paths are created using a cost-benefit ratio, and then are improved using four improvement procedures: 2-opt, Deletion-Insertion, 2-Exchange, and Location Swap.

The following steps describe the construction procedure. The value for $idletime_{(t,l)}$ is time between the arrival of the UAV at the location and the beginning of the time window; this value is zero if the UAV arrives after the beginning of the time window. The procedure is repeated for each UAV type, and produces set $P_{i,k}$, the set of ordered tasks for the path of UAV k of type i :

- (1) Obtain list of locations associated with tasks in S_i . This is set L_i .
- (2) Set $currentTime_i = 0$; current location is (t', l') .

(3) For each UAV of type i :

a. For each feasible tasks in L_i calculate:

$$\rho_{t,l} = \frac{value_{(t,l)}}{w_1 * obstime_t + w_2 * idletime_{(t,l)} + w_3 * late_t + w_4 * travelttime_{(t',l),(t,l),u}}$$

b. Find $\max\{\rho_{(t,l)}\}$; this is (t^*, l^*)

c. Add (t^*, l^*) to the set $P_{i,k}$.

d. Remove all other locations for task t from L_i .

e. Update $currentTime_k$. If $currentTime_k > endurance_i$ or $currentTime_k > horizon$, start path for next UAV; $k = k + 1$.

When completed, this procedure provides an initial path, in the form of an ordered set of locations, for each UAV. The improvement methods are then applied to the generated paths.

The first improvement procedure is modified from the Lin-Kernigan 2-opt method. In a 2-opt, two arcs on the path are replaced by two new arcs. If the change shortens the duration of the path and the path remains feasible, then the new arcs are incorporated into the path. This also changes the order of the tasks in the path.

The Deletion-Insertion method is an inter-path method; the method deals with two separate paths when attempting to improve the solution. The Deletion-Insertion method deletes a task from the first path, and then inserts the task into another path.

The 2-Exchange method is also an inter-path method similar to the Deletion-Insertion method. In the 2-Exchange method, two tasks are switched between two paths. This is equivalent to deleting two arcs in from two separate paths and replacing them with two new arcs.

The Location Swap Phase ensures that the tasks are completed in the most valuable location possible. In this phase, locations can be swapped out for higher value locations of the same task. For example, if the first task in Path A is currently performed at a location with a value of 10 but it can be performed at a location with a value of 15, then the higher value location is swapped for the lower value location, if feasible. The swap is considered for each location in each path.

After completing full path composite generation, the last step of COPA is to solve a linear program. The role of the linear program in the algorithm has two purposes. First, it ensures that the best set of generated path plans are chosen to create the observation plan for the system. Second, it ensures that the solution is a *set cover*, i.e., that each UAV in the system is assigned a path plan so that the solution utilizes each UAV.

The composite variable formulation utilizes the full path composite variables. This formulation models the UAV Operations Planning Problem and a solution contains the information needed for an operations plan.

The full path composite variables are used in a binary program for the UAV Operations Planning Problem. The following sets are used in the formulation:

$C = \text{Set of Composite Variables}$

$T = \text{Set of Tasks}$

$A = \text{Set of UAV Types}$

$Ua = \text{Set of UAVs of Type } a$

The formulation has a single binary decision variable that describes if composite c is included in the operations plan:

$$\gamma_c = \begin{cases} 1, & \text{if composite } c \text{ is included} \\ 0, & \text{otherwise} \end{cases}$$

Inputs for the formulation include information about the composites:

$$\begin{aligned} \delta_c^t &= \begin{cases} 1, & \text{if composite } c \text{ includes task } t \\ 0, & \text{otherwise} \end{cases} \\ \alpha_c^a &= \begin{cases} 1, & \text{if composite } c \text{ uses UAV type } a \\ 0, & \text{otherwise} \end{cases} \\ v_c &= \text{The value of composite } c \end{aligned}$$

The value of composite c , v_c , is found by summing the values of all tasks performed in the path included in composite c . With these inputs, the formulation for the *Full Path Composite Variable Binary Formulation* (FPCVBF) is:

$$\text{FPCVBF:} \quad \text{Maximize} \quad \sum_{c \in C} v_c \gamma_c \quad (4.1)$$

$$\text{Subject to} \quad \sum_{c \in C} \delta_c^t \gamma_c \leq 1 \quad \forall t \in T \quad (4.2)$$

$$\sum_{c \in C} \alpha_c^a \gamma_c \leq |U_a| \quad \forall a \in A \quad (4.3)$$

$$\gamma_c \in \{0, 1\} \quad \forall c \in C \quad (4.4)$$

Constraints 4.2 ensure that each task is performed only once in the solution. Constraints 4.3 allow for there to be a composite variable selected for each UAV in the system; essentially, the constraint ensures that the number of selected composites that use a UAV type is less than the number of UAVs of that type in the system.

The formulation given above is a binary program, because the decision variable γ_c is constrained to be in the set $\{0, 1\}$ in constraints 4.4. However, the constraints can be relaxed and still result in a binary solution. This is due to the structure of the problem; specifically, the structure of the constraints. The constraints create a feasible region with vertices where the values of the variables are either zero or one. Because of this, the relaxed version of the formulation provides an answer that is at the vertices of the feasible region. The reformulation is called the *Full Path Composite Variable Formulation*:

$$\text{FPCVF:} \quad \text{Maximize} \quad \sum_{c \in C} v_c \gamma_c \quad (4.5)$$

$$\text{Subject to} \quad \sum_{c \in C} \delta_c^t \gamma_c \leq 1 \quad \forall t \in T \quad (4.6)$$

$$\sum_{c \in C} \alpha_c^a \gamma_c \leq |U_a| \quad \forall a \in A \quad (4.7)$$

$$0 \leq \gamma_c \leq 1 \quad \forall c \in C \quad (4.8)$$

The result of the FPCVF is used to build the operations plan for the UAVs. The full paths whose composite variables take a value of 1 in the solution are combined to provide a path for each UAVs that spans the planning horizon. The combination of all selected paths provides a complete operations plan.

IV. Results and Discussion

In this section, the solutions and runtimes of COPA are compared to an implementation of the MIP to determine which is more suitable for solving the UAV Operations Planning Problem. The difference in the methods is that the COPA, which relies on a heuristic, might not provide an optimal solution. Conversely, the MIP is an exact solution method that provides the optimal solution.

The solution methods were implemented using computer software. COPA was implemented in Java, and utilizes CPLEX 11.0 to solve the FPCVF linear program. The MIP implementation also uses CPLEX 11.0, which employs a robust branch-and-bound method to solve the integer program.

Twenty pseudo-random test sets were created to analyze the solution methods. Four test sets were created each with the following number of tasks: 5, 10, 15, and 20. The tasks in each test set can be performed at two locations each. Therefore, for a test set with 5 tasks, there are 10 total locations, or, equivalently, 10 nodes in the network. Each test set includes two types of UAVs. Two of each type of UAV was included in each test set, for a total of four UAVs.

The runtimes of the COPA and MIP implementation are first compared to determine which is more suitable for solving the UAV Operations Planning Problem. The twenty test sets discussed in the previous paragraph were solved by both the COPA and MIP implementations. Each test set was solved three times using COPA. First, COPA generated 24 composites; second 48 composites were generated; and third, 72 composites were generated. Table 1 provides the runtimes for each test set.

<i>Size of Problem (Tasks, Locations)</i>	<i>COPA Runtime (s) 24 Comp.</i>	<i>COPA Runtime (s) 48 Comp.</i>	<i>COPA Runtime (s) 72 Comp.</i>	<i>MIP Runtime (s)</i>
(5,2)	0.81	1.17	1.376	2.93
(5,2)	0.734	0.817	0.938	20.53
(5,2)	0.613	0.526	0.848	7.42
(5,2)	0.617	0.679	0.99	16.28
(10,2)	0.357	0.98	0.913	3625
(10,2)	0.348	0.832	0.939	2640.53
(10,2)	0.398	0.456	0.49	6081
(10,2)	0.323	0.563	0.866	2732.16
(15,2)	0.82	1.099	1.163	7881.61*
(15,2)	0.825	0.941	1.088	1178.4*
(15,2)	0.414	0.626	0.764	19866.41*
(15,2)	0.439	0.772	1.23	220.65
(20,2)	0.764	0.748	0.989	4084.25*
(20,2)	0.457	0.736	1.537	2912.10*
(20,2)	0.543	1.095	1.211	40092.43*
(20,2)	0.48	0.757	1.087	21816.79*
* = Time until best solution was found (unable to prove).				

Table 1. Runtime Analysis.

The data presented in Table 1 indicates that, for each test set, COPA solves the problem in the shortest amount of time. Even when creating 72 composites, the average runtime for COPA was 1.027 seconds, compared to the average run time for the MIP of 28 minutes, 25.2 seconds, for the instances where an optimal solution was found. As the size of the test sets increased, the MIP was unable to confirm that the solution provided by the program was optimal, within the memory constraints of a desktop computer.

Another important aspect is the variance of the runtimes. In an operational situation, it would be detrimental to use an algorithm that could take significantly longer than expected; hence, the standard deviation of the runtime should be evaluated. Table 2 gives the standard deviations of the runtimes; this shows that the standard deviations of the runtimes for the MIP are orders of magnitude larger than the standard deviations of the runtimes for COPA.

<i>Size of Problem</i>	<i>Standard Deviation of COPA Runtime 24 Comp.</i>	<i>Standard Deviation of COPA Runtime 48 Comp.</i>	<i>Standard Deviation of COPA Runtime 72 Comp.</i>	<i>Standard Deviation of MIP Runtime</i>
(5, 2)	0.096	0.275	0.233	8.045
(10, 2)	0.031	0.241	0.210	1603.596
(15, 2)	0.229	0.205	0.206	9052.503
(20, 2)	0.140	0.174	0.239	17526.573

Table 2. Standard Deviation of Runtimes.

To compare the value of the operations plans developed by both methods, the optimal solutions provided by both solution methods were recorded for each test set. The analysis of the solutions was performed on the same 20 test sets as the runtime analysis. Similar to the runtime analysis, COPA was applied to each test set three times: first, 24 composites were generated; second, 48 composites were generated; and, third, 72 composites were generated.

The value of the operations plan developed by COPA depends on the number of composites generated. With a greater number of composites, the FPCVF linear program can choose from a wider selection of composites. Therefore, it is possible that the solution selected by COPA has higher value if more composites are generated. This is equivalent to stating that, by generating more composites, a larger solution space is explored by the algorithm in selecting the best solution.

The data in Table 3 indicates that COPA provided solutions with values that were close to the value of the optimal solution. Out of the nine cases where the MIP was able to confirm the optimality of the solution, COPA provided answers of equivalent value. The average optimality gap of the nine cases with confirmed optimal solutions; the average optimality gap was 3.95%. Another significant observation is that, in the test sets with 20 tasks, the computer was unable to provide a solution to the MIP that was better than the COPA solution before running out of memory.

Part of the motivation for developing COPA included the ability to solve large-scale problems in a reasonable amount of time. To understand how COPA handles large-sized problems, the algorithm was tested using test sets with an increasing number of tasks. Each test set had 15 UAVs, five of one type and ten of another type. The planning horizon for the test set was six hours.

<i>Size of Problem</i>	<i>Value of COPA Solution 24 Comp.</i>	<i>Value of COPA Solution 48 Comp.</i>	<i>Value of COPA Solution 72 Comp.</i>	<i>Value of MIP Solution</i>	<i>Lowest Optimality Gap</i>
(5,2)	245	255	255	295	13.56%
(5,2)	161	161	161	161	0%
(5,2)	274	275	275	281	2.14%
(5,2)	279	279	279	279	0%
(10,2)	543	543	543	555	1.30%
(10,2)	660	660	664	739	10.15%
(10,2)	634	696	696	696	0%
(10,2)	432	434	434	466	6.87%
(15,2)	851	888	888	959*	2.10%
(15,2)	884	884	884	884*	0%
(15,2)	920	960	960	992*	3.23%
(15,2)	959	959	959	974	1.54%
(20,2)	1207	1247	1265	703*	**
(20,2)	1102	1102	1102	784*	**
(20,2)	1138	1171	1190	1127*	**
(20,2)	1122	1125	1139	766*	**

* = Best solution; computer unable to prove
** = Algorithm solution greater than MIP solution

Table 3. Optimality Comparison.

The testing began with 100 tasks that could be performed at 10 locations each; therefore, the number of nodes in the test set was 1,000. Then, 100 tasks were added for each test set. The largest test set included 500 tasks that could be performed at 10 locations each for a total of 5,000 nodes. During testing, COPA generated 30 composites for each test set.

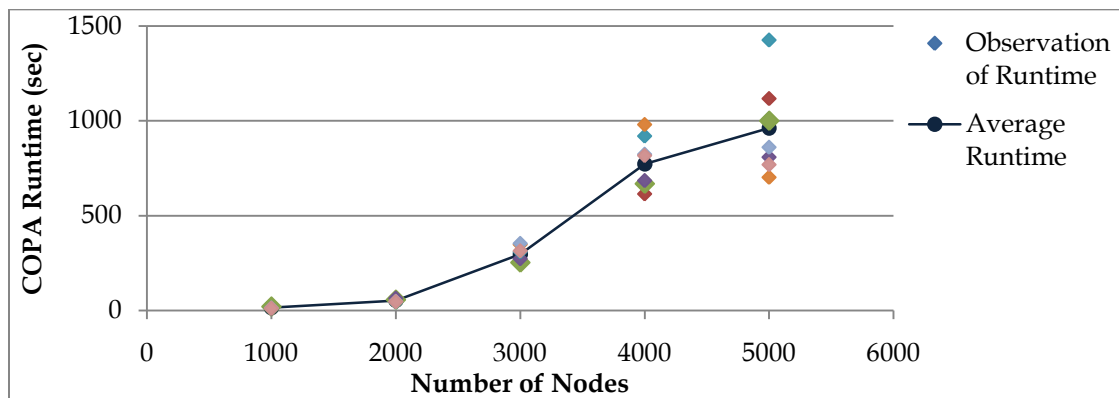


Figure 2. COPA Runtimes for Large-Scale Instances of the UAV Operations Planning Problem.

Figure 2 shows the runtimes of the large test sets; the runtimes are displayed for eight test sets of each size. COPA was able to handle these large test sets in a reasonable amount of time. The longest runtime was 23 minutes and 46.4 seconds for a test case with 500 tasks that could be performed at 10 locations each. This runtime is reasonable for an operational situation with planning horizon of six hours.

In conclusion, COPA is suitable for solving the UAV Operations Planning Problem. The MIP implementation of the UAV Operations Planning Problem provided the optimal solution for the operations planning problem. However, for cases with 40 nodes, it was difficult for optimization software to determine an optimal solution. It was shown that COPA is able to solve problems with fewer than 40 nodes in less than two seconds, with an average optimality gap of 3.95%. The COPA algorithm is capable of solving large-scale problems with up to 5000 nodes in under 30 minutes on a desktop computer.

References:

- [1] Armacost, A., Barnhart, C., and Ware, K., "Composite Variable Formulations for Express Shipment Service Network Design," *Transportation Science* 26, 2002.
- [2] Boussier, S., Feillet, D., and Gendreau, M., "An Exact Algorithm for Team Orienteering Problems," *A Quarterly Journal of Operations Research* 5(3), 211-230, 2006.
- [3] Gendreau, M., Hertz, A., Laporte, G., and Stan, M., "A Generalized Insertion Heuristic for the Traveling Salesman Problem with Time Windows," *Operations Research* 46(3), 330-335, 1998.
- [4] Golden, B. L., Levy, L., and Vohra, R., "The Orienteering Problem," *Naval Research Logistics* 37, 307-318, 1987.
- [5] Miller, J. V., "Large-Scale Dynamic Observation Planning for Unmanned Surface Vessels," Masters Thesis, Massachusetts Institute of Technology, 2007.
- [6] Ramesh, R. and Brown, K. M., "An Efficient Four-Phase Heuristic for the Generalized Orienteering Problem," *Computers and Operations Research* 18, 151-165, 1991.
- [7] Ramesh, R., Yoon, Y., and Karwan, M. H., "An Optimal Algorithm for the Orienteering Tour Problem," *ORSA Journal on Computing* 4, 1992.
- [8] Ropke, S., Cordeau, J., and Laporte, G., "Models and Branch-and-Cut Algorithms for Pickup and Delivery Problem with Time Windows," *Networks* 49(4), 258-272, 2007.
- [9] Tsiligrides, T. "Heuristic Methods Applied to Orienteering." *Journal of the Operational Research Society* 35, 797-809, 1984.