CSCE 686 Complex Optimization Algorithm Design Project

1Lt Chip Van Patten*
Air Force Institute of Technology
Dayton, Ohio 45433
Email: *donald.vanpatten@afit.edu

***Abstract*—This project addresses a problem in the field of Unmanned Aircraft Systems (UASs) dealing with assigning UAS orbits. Given a limited number of UAVs, each based in specific locations – each with specific ranges of flight and operating costs – and a list of targets to surveil, how do we assign surveillance orbits to the UASs such that all targets are surveilled while keeping the UASs close enough to their base of operations that they may safely return, either when their mission finishes or they've exhausted their fuel, at minimum cost? The project will attempt to integrate this application domain with the set cover problem domain and a number of algorithm domains to solve the problem efficiently.**

I. Introduction

This project addresses a problem in the field of Unmanned Aircraft Systems (UASs) dealing with assigning UAS orbits to cover a list of targets. Given a limited number of UAVs, each assigned to specific units based in specific locations – each with specific ranges of flight and operating costs – and a list of targets to surveil, how do we assign surveillance orbits to the UASs such that all targets are surveilled while keeping the UASs close enough to their base of operations that they may safely return, either when their mission finishes or they've exhausted their fuel, at minimum cost? The project will attempt to integrate this application domain with the set cover problem domain and a number of algorithm domains to solve the problem efficiently.

The rest of this paper is organized as follows. Section II, discusses the design process starting with the problem domain (PD) and algorithm domain (AD) and ending with a coded implementation. In Section II-B, the problem domain of the UAS problem discussed briefly in this section is revisted and discussed in greater detail. Section IV describes and analyzes the experiments run. Section V concludes with a summary of the project and brief discussion of related works.

II. CSCE 686 Project Algorithmic Design Process

This section details the algorithmic design process using the top down disciplined design approach specified in [12], [11]. The problem domain is discussed in Section II-A. Integration of the problem domain with the application domain is detailed in Section II-B. The algorithm domain is discussed Section III where the initial integration of the algorithm and problem domains begins. Section III-D refines the data structures from III and integrates them into the program design. Pseudocode for the algorithm is presented in Section III-E and, finally, an implementation in code based on the aforementioned pseudocode is described in Section III-F.

A. Problem Domain

1) Problem Domain Discussion: The set cover problem is a class of problems in which one seeks to select a minimum number of sets, at minimum cost, so that the union of all selected sets contains all elements in the input [16]. More formally, given a universe, or ground set of elements, $\mathcal{R} = \{r_1, \ldots, r_m\}$ and a family of sets $\mathcal{S} = \{S_1, \ldots, S_N\}$, a set covering of $\mathcal{R}$ exists if

$$\bigcup_{i=1}^{k} S_{j_i} = \mathcal{R},$$

with S_{j_i} being the covering sets [5, pp. 39-40]. Christofides further state that if, for each $S_j \in \mathcal{S}$, "there is associated a (positive) cost c_j" [5, p. 39], then the goal of the set cover problem is to find a set of sets $\mathcal{S}'$ which includes all m elements of $\mathcal{R}$ while minimizing $\sum_{i=1}^{k} c_j$ [5], [16], [18], [22].

In the classic, or un-weighted, set cover problem, all $c_j = 1$ [16]. In this paper we will only deal with a weighted variant of the set cover problem, consequently, for the purposes of this paper, the term *set cover problem* is synonymous with *weighted set cover problem*, but the former will be used exclusively.

2) Symbolic Set Development and Discussion: We will now develop a symbolic set (adapted from [5], [11]) which will formalize the notation used throughout this paper and, hopefully, aid in understanding and clarity as we progress in our development to the algorithm and application domains.

Domains:

$D_i : (\mathcal{R}, \mathcal{S})$
$\mathcal{R}$: set of m elements r
$\mathcal{S}$: set of N sets with costs c that may cover $\mathcal{R}$
set cover : $\bigcup_{i=1}^{k} S_{j_i} = \mathcal{R}$ [5, Eq. 3.12]
– union of sets cover $\mathcal{R}$ elements
D_o : set of set-covers $\mathcal{S}'$ of $\mathcal{R}$ with additive cost

3) $0/1$ Problem Description: An integer linear program is a special case of integer programming "in which unknowns are binary, and only the restrictions must be satisfied" [21].

According to [5], the set cover problem "can be formulated as a $[0/1]$ linear program" as follows:

$$\text{minimize} \sum_{j=1}^{N} c_j \xi_j$$
$$\text{subject to} \sum_{j=1}^{N} t_{ij} \xi_j \geqslant 1 \qquad \forall r_i \in \mathcal{R}$$

where $c_j \geqslant 0$, $\xi_j \in \{0, 1\}$, if $S_j \notin \mathcal{S}'$ or $S_j \in \mathcal{S}'$ respectively, and $t_{ij} \in \{0, 1\}$, if $r_i \notin S_j$ or $r_i \in S_j$ respectively [5, p. 39-40].

This is different from the set cover problem defined above in that we encode, using binary representation, whether every set is in the set cover or not. We assign a 1 if the set is in the set cover, and a 0 otherwise. This provides a $\log n$-approximation algorithm for the minimum set cover problem [22].

4) Problem Domain Complexity: In 1972, Karp, et al. proved that the set cover problem is NP-Complete [9], while the optimization version is NP-Hard [22]. According to [22], [11], the complexity of the set cover problem is the powerset of $\mathcal{S}$, that is $\mathcal{O}(2^{|\mathcal{S}|}) \rightarrow \mathcal{O}(2^N)$, for both the search and solution spaces. This is due to the fact that a solution is found, if it exists, only by searching the space of all possible combinations of $\mathcal{S}$.

5) Problem Domain Specification: Using the symbolic notation developed in Section II-A2 we define the input and output domains and conditions for the set cover problem domain.

D_i : $(\mathcal{R}, \mathcal{S})$
$\mathcal{R}$: set of m elements r
$\mathcal{S}$: set of N sets with costs c that may cover $\mathcal{R}$
set cover : $\bigcup_{i=1}^{k} S_{j_i} = \mathcal{R}$ [5, Eq. 3.12]
– union of sets cover $\mathcal{R}$ elements
D_o : set of set-covers $\mathcal{S}'$ of $\mathcal{R}$ with additive cost

Solution:

Minimize additive cover costs subject to following constraints:

- set cover:
$\sum_{j=1}^{N} t_{ij} \xi_j \geqslant 1, \quad i = 1, 2, \ldots, M$ [5, Eq. 3.14]
cover all elements at least once

B. Application Problem Domain

1) Application Problem Domain Discussion: The UAS problem deals with assigning UAS orbits to cover a list of targets. Given a limited number of UAVs, each assigned to specific units based in specific locations – each with specific ranges of distance able to travel and operating costs – and a list of targets to surveil, how do we assign surveillance orbits to the UASs such that all targets are surveilled while keeping the UASs close enough to their base of operations that they may safely return, either when their mission finishes or they've exhausted their fuel, at minimum cost? The UAS problem relates to the set cover problem in that we are trying to cover a set of targets with a set of UAVs based on their locations, all while keeping them within a specified distance of their home stations and minimizing the associated operating costs.

Formally, there exists a universe of targets that we want to surveil $\mathcal{T} = \{t_1, \ldots, t_m\}$, each with a specific hideout location t_{i_h}, a set of UASs $\mathcal{U} = \{u_1, \ldots, u_n\}$, each with an assigned squadron and operating cost, a set of squadrons $\mathcal{F} = \{f_1, \ldots, f_q\}$ with specific geographic locations $f_{j\ell}$, and, finally, a set of schedules $\mathcal{A} = \{a_1, \ldots, a_r\}$ for each target which denote time windows in which the target is able to be surveilled. The objective is to surveil each target t during their time window a using a minimum number of UASs $u \in \mathcal{U}$ at minimum cost c such that each target is surveilled.

A review of the available literature reveals that, while much research has been performed on routing UASs and target coverage, none of the available sources provide insight into the constrained problem defined in this section. The most similar research can be found in [8], which uses a Max-Min Ant System algorithm, a member of the ant colony optimization algorithm family [20]. Other research with related, but not similar, goals to this project are discussed in [1], [3], [4], [14], [15], [17].

2) Application Problem Domain Complexity: Similar to the problem domain complexity described in Section II-A4, the complexity of the application domain is the powerset of each search space $\mathcal{O}(2^{|\mathcal{T}|} \times 2^{|\mathcal{U}|} \times 2^{|\mathcal{F}|} \times 2^{|\mathcal{A}|}) \rightarrow \mathcal{O}(2^{m+n+q+r})$, or every possible combination of targets, UASs, squadrons, and surveillance time windows.

We now have to prove that the application domain, the UAS problem, is NP-Complete. This can be shown using the technique described in [10] in which the constraints described in Section II-B1 are relaxed to facilitate a polynomial-time transformation into the form of a known NP-Complete problem. This would allow a polynomial-time black-box solver for the known NP-Complete problem, if one existed, to solve the relaxed-constraint UAS problem. The UAS problem with relaxed constraints could then be said to be at least as hard as NP-Complete.

Relaxing the constraints of the UAS problem results in a problem as follows: given a universe, or set of targets, $\mathcal{R}$ a set of UASs $\mathcal{U}$, and a family of sets $\mathcal{S}$, containing UASs available to surveil specific targets. Assign to each $s \in \mathcal{S}$ an operating cost c. The goal is then to surveil each target in $\mathcal{R}$ at minimum cost. This transformation can be done in polynomial time.

The solution returned is easily verified by iterating through $\mathcal{R}$ and checking that there is in fact a UAS $\in \mathcal{U}$ assigned to surveil that target. The complexity of the verification is $\mathcal{O}(|\mathcal{R}| \times |\mathcal{U}|)$

Therefore, since we have shownn that, in polynomial-time, the UAS problem can be transformed to the set cover problem (a known NP-Complete problem), $UAS \leqslant_p Set\ Cover$, and we can verify in polynomial-time the solution, the UAS problem is NP-Complete.

III. Algorithm Domain/Problem Domain Initial Integrations

A. Selection of Deterministic Algorithm

The deterministic algorithm used for this project is the A* search algorithm. The heuristic function will be defined the same way the classic greedy approach to this problem is as per [22], with the exception that the heuristic is inverted, as done in [7], due to the fact that A* chooses the next search node which has the $h(n)$ closest to 0. The heuristic function developed for this project is defined as: $h(n) = -\frac{\text{\# of targets surveilled}}{\text{cost}}$. The function for deciding which node to visit next is:

$$\text{minimize} f(n) = h(n)c(n, n').$$

1) Problem Domain/Algorithm Domain Developing Specification:

2) Pseudocode: Name: SCP/SPP A* Search

Algorithm 1 is a version of the A* algorithm detailed in [13], modified to delay termination to find a complete solution and to include the heuristic detailed in Section III-A.

B. Selection of Stochastic Algorithm

For the simulated annealing

"This theorem states that no search algorithm is better than a random search on the space of all possible problems in other words, if a particular algorithm does better than a random search on a particular type of problem, it will not perform as well on another type of problem, so that all in all, its global performance on the space of all possible problems is equivalent to a random search." [6], [19]

1) Algorithm/Program Design Methodology:

2) Refinement of PD/AD with Data Structures:

C. Algorithm Design Specifications

1) Integration of Problem and Algorithm Domains:

D. Intermediate Algorithm Designs

E. Algorithm Pseudocode

F. Program Implementation

The implementation for this project (along with the LaTeX files for this document) can be seen on GitHub[1].

IV. Design of Experiments

Experiments for this project were designed in accordance with the guidelines set forth in [2]. An Apple MacBook Pro equipped with a 2.8 GHz Intel Core i7 processor with 16 GB of 1600 MHz DDR3 RAM running Mac OS X version 10.11.4 was used to run all experiments.

[1]https://github.com/chipvp/686Project

Algorithm 1 A* Search Algorithm from [13] modified with a heuristic and delayed termination to solve SCP/SPP

/* *Initialization* */
generate tree
$OPEN$: priority queue to hold unvisited nodes
$CLOSED$: set to hold visited nodes
n : a set of sets with properties id, cost, and vector of children

Put the start node s on $OPEN$
while true **do**
/* *Solution Function* */
/* *Delayed termination* */
  if all elements covered && $OPEN$ is empty **then**
    return solution obtained by tracing back the pointers from n to s
  end if
/* *Selection Function* */
  Remove from $OPEN$ and place on $CLOSED$ a set for which $f(n) = h(n) + c(n, n')$ is minimum
/* *Next-State-Generator Function* */
  Otherwise expand n, generating all its successors, and attach to them pointers back to n.
/* *Feasibility Function* */
  for each successor n' of n **do**
/* *Objective Function* */
    if n' is not already on $OPEN$ or $CLOSED$ **then**
      Estimate $h(n) = -\frac{\text{\# of elements in the set}}{\text{cost}}$
      Calculate $f(n') = h(n') + c(n, n')$
    else if n' is already on $OPEN$ or $CLOSED$ **then**
      Direct its pointers along the path yielding the lowest $h(n')$
    end if
    if n' required pointer adjustment && was on $CLOSED$ **then**
      Put n' on $OPEN$
    end if
  end for
end while

V. Conclusions

References

[1] G. Avellar, G. Pereira, L. Pimenta, and P. Iscold. Multi-UAV Routing for Area Coverage and Remote Sensing with Minimum Time. *Sensors*, 15(11):27783–27803, 2015.

[2] R. Barr, B. Golden, J. Kelly, W. Steward, and M. Resende. Guidelines for designing and reporting on computational experiments with heuristic methods. In *Proceedings of International Conference on Metaheuristics for Optimization, Kluwer Publishing*, pages 1–17, 2001.

[3] C. Caillouet. Energy efficient point coverage problem using UAVs.

[4] W. Carlyle, J. Royset, and R. Wood. Routing military aircraft with a constrained shortest-path algorithm. Technical report, DTIC Document, 2007.

[5] N. Christofides. *Graph Theory: An Algorithmic Approach*. Computer Science and Applied Mathematics. Academic Press, Incorporated, 1975.

[6] P. Collet and J.P. Rennard. Stochastic optimization algorithms. *arXiv preprint arXiv:0704.3780*, 2007.

[7] P. Jordan and C. Van Patten. CSCE 686: Homework 8. (Unpublished class homework). Air Force Institute of Technology, May 2016.

[8] M. Karakaya. UAV Route Planning For Maximum Target Coverage. *CoRR*, abs/1403.2906, 2014.

[9] Richard M. Karp. *Complexity of Computer Computations: Proceedings of a symposium on the Complexity of Computer Computations, held March 20–22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, and sponsored by the Office of Naval Research, Mathematics Program, IBM World Trade Corporation, and the IBM Research Mathematical Sciences Department*, chapter Reducibility among Combinatorial Problems, pages 85–103. Springer US, Boston, MA, 1972.

[10] J. Kleinberg and E. Tardos. *Algorithm Design*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2005.

[11] G. Lamont. CSCE 686 - Advanced Algorithm Design: SCP/SPP Global Depth-First-Search Back-Tracking Design. 2016. (Unpublished class notes). Air Force Institute of Technology.

[12] G. Lamont. CSCE 686 - Advanced Algorithm Design: Specific Disciplined Design Approach. 2016. (Unpublished class notes). Air Force Institute of Technology.

[13] J. Pearl. *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. The Addison-Wesley Series in Artificial Intelligence. Addison-Wesley, 1984.

[14] S. Quintero, F. Papi, D. Klein, L. Chisci, and J. Hespanha. Optimal UAV coordination for target tracking using dynamic programming. In *49th IEEE Conference on Decision and Control (CDC)*, pages 4541–4546, December 2010.

[15] E. Semsch, M. Jakob, D. Pavlíček, and M. Pěchouček. Autonomous UAV surveillance in complex urban environments. In *Web Intelligence and Intelligent Agent Technologies, 2009. WI-IAT'09. IEEE/WIC/ACM International Joint Conferences on*, volume 2, pages 82–85. IET, 2009.

[16] T. Stern. Set Cover Problem (Chapter 2.1, 12). (Unpublished class notes). Massachussetts Institute of Technology: Seminar in Theoretical Computer Science, February 2006.

[17] K. Sundar and S. Rathinam. Algorithms for Routing an Unmanned Aerial Vehicle in the presence of Refueling Depots. *CoRR*, abs/1304.0494, 2013.

[18] E. Talbi. *Metaheuristics: From Design to Implementation*. Wiley Publishing, 2009.

[19] Wikipedia. No free lunch theorem — Wikipedia, The Free Encyclopedia, 2015. [Online; accessed 20-April-2016].

[20] Wikipedia. Ant colony optimization algorithms — Wikipedia, The Free Encyclopedia, 2016. [Online; accessed 2-May-2016].

[21] Wikipedia. Integer programming — Wikipedia, The Free Encyclopedia, 2016. [Online; accessed 2-May-2016].

[22] Wikipedia. Set Cover Problem — Wikipedia, The Free Encyclopedia, 2016. [Online; accessed 25-April-2016].