Bi-Objective Study for the Assignment of Unmanned Aerial Vehicles to Targets

Ryan D. Friese*, James A. Crowder§, Howard Jay Siegel†ǧ, and John N. Carbone¥£

*Pacific NW Nat. Lab
Richland, WA, 99354
U.S. Dept. of Energy
Washington DC, 20585, USA

§Colorado Engineering, Inc.
Colorado Springs, CO 80920
USA

†Elect. & Computer Engineering Dept.
ǧComputer Science Dept.
Colorado State, University
Ft. Collins, CO 80523, USA

¥Elect. & Computer Engineering Dept.
£Computer Science Dept.
Southern Methodist University
Dallas, TX 75205, USA

Email: Ryan.Friese@pnnl.gov, jim.crowder@coloradoengineering.com, HJ@colostate.edu, John.Carbone@forcepoint.com

Abstract— Historically, research shows that multi-vehicle, multi-constraint, surveillance problems require a combinatorial optimization solution. In many of these surveillance missions, the overall objective is to provide plans for surveillance tasks for UAVs (unmanned aerial vehicles) visiting, or "surveilling," targets across geographically distributed areas. Surveillance plans are created with the goal of maximizing the number of targets the fleet of UAVs can surveil in a given period of time (in this paper, 24 hours) under a given set of constraints related to total energy usage (the energy available to each UAV).. Here, we present a bi-objective task planning genetic algorithm (GA) that provides a Pareto set of near optimal surveillance plans, given the above conflicting energy and surveillance objectives. Future work will expand these algorithms to support multi-objective mission planning, including speed, distance, weather conditions, and other factors that would affect the overall surveillance opportunities.

Keywords—bi-objective optimization, genetic algorithms, Pareto fronts, surveillance, UAVs

1. INTRODUCTION

Surveillance planning for multi-vehicle, multi-target, ad-hoc tasking represents a real-world situation for active battlefield scenarios [1]. Figure 1 below illustrates an example scenario. In this scenario, we have seven UAVs, flying in two patterns forming concentric circles around the geographic area to be surveilled. Within the geographic area to be surveilled, we have identified targets to be surveilled by the UAVs. Each UAV may have different sensors on board and each target may have different priorities and need to be surveilled by a given sensor type (e.g., IR, SAR).

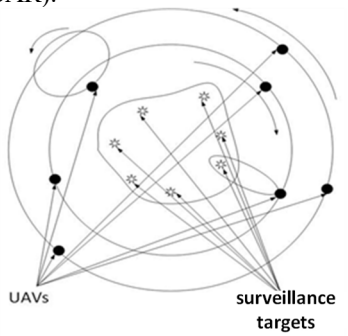


Figure 1: Example of multi-vehicle, multi-target surveillance problem

UAV/drone planning in these cases can often be an oversubscribed, heterogeneous scheduling problem, because UAV surveillance tasks arrive dynamically, and the UAV planner must map tasks to UAVs for mission execution. The number of surveillance planning requests at any moment in time may exceed the available number of UAVs, making the planning problem oversubscribed. It is possible that each UAV may have different capabilities and different physical constraints and characteristics, making the real-time, mission planning problem an oversubscribed, heterogeneous planning problem [2].

Examples of past UAV research include cooperative task assignment and path planning of multiple UAVs [3], three-dimensional offline path planning using a multi-objective evolutionary algorithm [4], and offline/online path planning for UAV navigation [5]. However, here we describe a bi-objective genetic algorithm for a surveillance value performance measure and resource allocation constraints required to manage and successfully plan a set of regularly scheduled surveillance requests. In addition, if conditions change, as situational awareness information is obtained, the surveillance schedule could be re-computed [6]. Each mission task has a weight (importance) that will be utilized to measure success/failure of the planning algorithms. The results will illustrate the benefits of the algorithms and constraints for making allocation decisions in an oversubscribed, real-time planning environment [7].

Such multi-vehicle, multi-constraint, surveillance planning problems are combinatorial optimization problems in which the overall objective is to schedule a set of optimized surveillance planning requests that provides a plan for UAVs visiting, or "surveilling" over geographically distributed targets within a given area. Surveillance plans are created with the goal of maximizing the number of targets the fleet of UAVs can surveil in a given period of time (in this paper, 24 hours) under a given set of constraints related to total energy usage (the energy available to each UAV) [8]. Here, we present a bi-objective task planning genetic algorithm (GA) that provides a Pareto set of near optimal surveillance plans, given the above conflicting energy and surveillance objectives. The contributions of this paper include:

1. Defining a surveillance value performance measure that includes the targets surveilled, the number of times each target is surveilled, and the priority of each target;

2. Design of a model for surveilling heterogeneous targets by heterogeneous UAVs;
3. Implementation of a genetic algorithm for a bi-objective study of energy versus surveillance performance for a set of realistic system parameters;
4. The construction of a Pareto front of near optimal mappings of targets to UAVs to use to study tradeoffs between those two objectives.

The paper is organized as follows. First, a description of the bi-objective optimization problem is presented with tables describing the UAV and target characteristics in Section 2. Section 3 provides a detailed description of the genetic algorithm that was used for the bi-objective surveillance plan optimization through the generation of the Pareto fronts. In Section 4, we present the results illustrating the creation of a surveillance plan for the UAVs to surveil the targets, using the bi-objective function with total energy usage and weighted number of surveilled targets as the objectives. Section 5 discusses related work. We conclude in Section 6 and present some ideas for future work.

2. PROBLEM DESCRIPTION

For purposes of this paper, we are limiting our optimization to two parameters: total energy usage and total number of surveils of the targets, weighted by the target's priority. The tables below show the parameters we consider in this paper.

Table 1 illustrates the constraints associated with UAV fleet. For our example mission planning/scheduling problem, we assume seven UAVs. Table 1 shows the total energy (normalized to a maximum of 1 for the UAV with the most energy) available to a given UAV for surveillance of tasks over a 24-hour period. It is assumed that there is enough reserve energy allotted to each UAV to return to a "home" for refueling/recharging once the nominal energy is at 0. Also shown is the energy cost/hour for operating each UAV, along with the sensor types available for each UAV.

Table 1: UAV Characteristics

UAV characteristic	UAV #1	UAV #2	UAV #3	UAV #4	UAV #5	UAV #6	UAV #7
total energy (0-1)	1	0.8	0.9	0.7	0.8	0.6	1
energy cost/hour (0-1)	0.08	0.08	0.08	0.1	0.125	0.07	0.15
sensor type	VIS & IR	VIS	SAR	UV & IR	SAR & IR	VIS & IR	SAR & UV

Table 2 describes the characteristics of the targets to be surveilled by the UAVs. For our example problem for this paper, there are nine targets shown. Each target has a given surveillance priority, the surveillance time required for each surveil of that target, the sensor types that may be used for each target surveillance, and the maximum number of surveils per day desired for each target.

Table 2: Target Characteristics

target characteristics	target #1	target #2	target #3	target #4	target #5	target #6	target #7	target #8	target #9
proritiy	1	2	6	7	5	4	3	3	1
surveillance time - hrs	3	4	1	3	2	1	3	3	4
surveillance type	SAR/IR/VIS	SAR/UV	VIS/SAR	Any	VIS	IR/UV	SAR/VIS	SAR/IR/VIS	VIS
surveil fequency #/day	4	3	5	2	4	5	3	3	4

Figure 2 illustrates this distance normalization. For the UAV sensors, each sensor type is:

- **VIS**: visible light spectrum sensor
- **IR**: infrared spectrum sensor
- **UV**: ultraviolet spectrum sensor
- **SAR**: surface acoustical radar sensor

As shown in Figure 2, a circle is drawn, encompassing the target area, designated with radius 1. The UAVs stay in concentric circle flight paths around the target area waiting for tasks (surveillance missions). The two flight paths are normalized and are determined, based on the normalization of the given geographic area to a radius of 1, such that any given UAV can surveil any given target from within its concentric flight path.

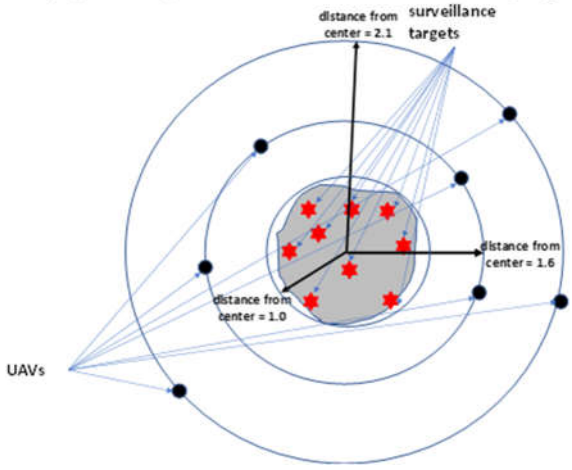


Figure 2: Surveillance example with normalized distances

The overall performance measures for the planning algorithms are:

1. Minimize the overall normalized absolute energy, summed over all the UAVs (see Table 1)
2. Maximize weighted total number of targets surveilled based on priority, i.e., for each time (up to the allowed maximum) a target complete is surveilled for the required amount of time (see Table 2) the system performance is incremented by that targets priority.

Any operations still occurring when a period ends will get a proportional credit, based on the completed time for that surveil (see Table 2). Table 2 describes the time a UAV must spend on the target to have a complete surveillance. The planning algorithm can repeat a given target with the same or different UAV up the number of times to be surveilled (see Table 2). Surveils to the same target can be continuous in time or spaced out temporally. For this study, we do not consider the distance between target and UAV, so each UAV can surveil any target from wherever the UAV is within its flight plan (concentric circles around the geographic area.

3. MISSION PLANNING ALGORITHMS

3.1 Genetic Algorithms

Genetic algorithms (GAs) are common evolutionary optimization techniques useful in solving problems that contain large and complex search spaces [9, 10, 11, 12, 13]. GAs try to emulate the process of natural selection; i.e., producing better (fitter) solutions as time progresses. A typical GA maintains a population of individuals called chromosomes. Each chromosome is a solution to the problem being solved. Chromosomes are compared with one another by evaluating their fitness. Fitness functions are often, but not always, the objective function to optimized. Chromosomes are further composed of genes, the base component of a solution, and their representation is highly dependent on the problem being solved. Our paper implements a popular multi-objective GA, the NSGA-II [14] (described below).

Better solutions in a GA are produced as the population evolves through time. Evolution occurs using three genetic operators: selection, crossover, and mutation. During selection, chromosomes are chosen as parents to "mate" and produce offspring chromosomes. Typically, selection operators are biased towards selecting more fit chromosomes. The crossover operation takes the chromosomes chosen during selections and swaps a portion of the genes of each parent into one another, resulting in offspring chromosomes that contain genetic information from both parents. Finally, mutation operates on chromosomes individually, with individual genes in a chromosome being randomly mutated to introduce new genetic information. Selection, crossover, and mutation are applied to the population until some stopping criteria is met, e.g., the population converges, or a given number of iterations have been performed.

3.2 NSGA-II

The NSGA-II is a popular bi-objective genetic algorithm developed in 2002 and is used for a variety of optimization problems [14]. The basic algorithm flow is illustrated in Figure 3 [14]. The NSGA-II utilizes solution dominance to produce Pareto fronts, i.e., the set of solutions that are not dominated by other solutions. Figure 4 illustrates the principle of solution dominance. In the figure, A dominates D because A is better in both objectives, any solution below and to the right of A **would** be dominated by A, while any solution above and to the left of A would dominate it. None of A, B, or C dominate each other because each one out performs the others in at least one objective, e.g., B consumes less energy than A and C, but A has a higher value than B and consumes less energy than C.

3.3 Chromosome Structure

For our use, as described in Section 2, we are trying to solve surveillance of multiple targets using multiple UAVs, which we formulate as a resource allocation (task scheduling) problem. Specifically, we equate the targets to tasks that need to be executed with the UAVs as the resources [7]. Thus, a solution consists of the allocation of UAVs to surveillance targets.

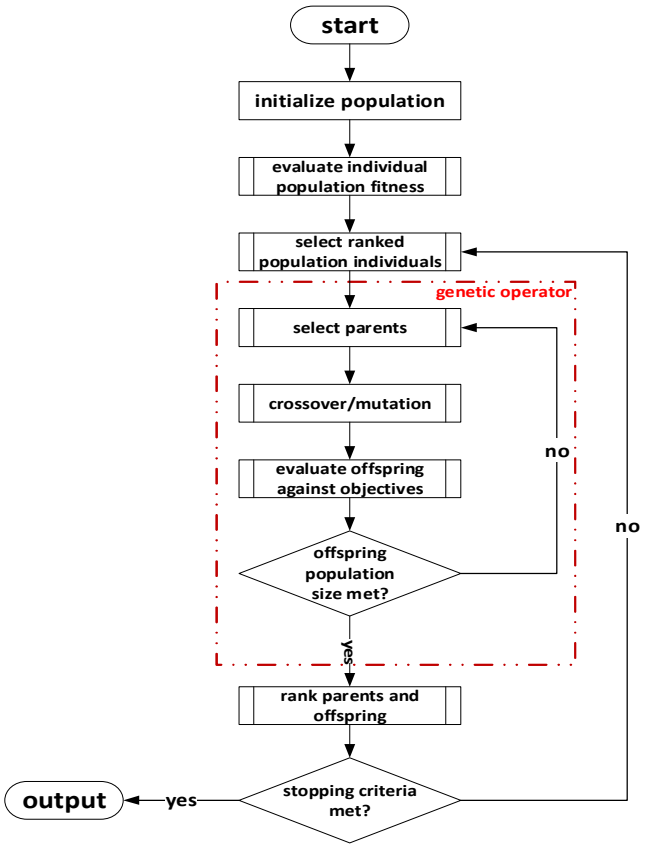


Figure 3: The NSGA-II algorithm flow

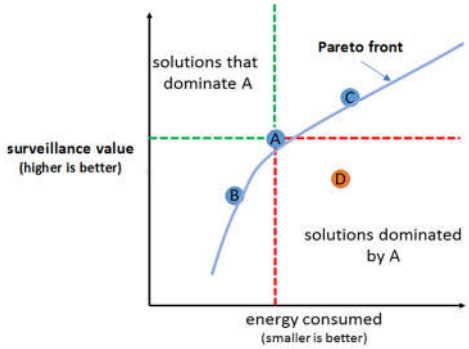


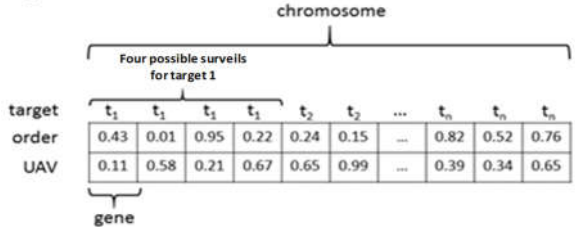
Figure 4: Example of dominance for four solutions: A, B, C, and D. Solution A dominates D because A has lower energy consumption as well and a higher surveillance value. None of A, B, or C dominate each other because they all outperform one another in one objective, but not both.

Given the assumptions and constraints of the problem, we have defined that each target has a maximum number of surveils for which value can be earned. Thus, at its most basic representation an individual gene within a chromosome represents a possible surveil for each target. To fully encode a solution, each gene contains two floating-point numbers in the range from 0 to 1, representing:

1. A global scheduling order where the lower the number, the earlier the associated target will be surveilled.

2. The UAV that will be used to perform the surveillance.

Figure 5 illustrates a sample chromosome, with the entries ranging from 0 to 1.



target	t_1	t_1	t_1	t_1	t_2	t_2	...	t_n	t_n	t_n
order	0.43	0.01	0.95	0.22	0.24	0.15	...	0.82	0.52	0.76
UAV	0.11	0.58	0.21	0.67	0.65	0.99	...	0.39	0.34	0.65

Figure 5: Chromosome structure

Floating point numbers are utilized because they allow for every chromosome to completely encode a valid solution without requiring expensive "fix" operations during the crossover or mutation operations. In practice, this applies to each data field for different reasons. For target surveillance ordering, it is highly unlikely that any two genes will contain precisely the same floating-point value, thus we can always find an exact ordering amongst the genes. In the UAV representation, utilizing floating-point numbers allows us to shift the actual calculation of the UAV from the crossover and mutation operations into the fitness evaluation of each chromosome, allowing the crossover and mutation functions to be very fast.

Furthermore, representing these values as double-precision means it will be highly unlikely that we will have two identical chromosomes within a given solution space, allowing us to maintain more genetic diversity within the populations. Finally, storing the values as double-precision numbers allows us to use more exotic mutation and crossover operations, which will be useful in later work as we expand the problem space and increase the number of objectives.

3.4 Chromosome Fitness Evaluation

Given the structure of our chromosomes, we have implemented an efficient evaluation function to construct the actual resource allocation and then to calculate fitness in terms of value earned and energy consumed. Evaluation of a chromosome required adherence to several constraints:

1. Each target has a maximum number of useful surveils (inherently handled by the structure of the chromosome itself).
2. The constraint that only one UAV can surveil a target at a time is handled by the order value within a gene and by a vector (of size number of target types) that contains the available time for each target (Algorithm 1, line 3).
3. A vector of UAV available times (Algorithm 1, line 1) represents when a new surveil can occur and prevents individual UAVs from surveilling multiple targets simultaneously.
4. UAVs are constrained by the amount of energy they contain; thus, we create a vector that maintains the current amount of energy consumed for each UAV (Algorithm 1, line 2).

The fitness evaluation algorithm is shown in Figure 6. The first step of the algorithm is to sort the genes, based on their order values, and then iterate over the genes in descending order (Algorithm 1, line 5). Both the target and UAV are extracted from the gene (Algorithm 1, lines 6,7). The target is determined based on the location of the gene in the chromosome. To calculate the specific UAV for a gene, we divide 1 by the number of possible UAVs (based on sensor constraints in Table 1). This creates equally sized buckets from 0 to 1 that map to the possible UAV's for each target. The UAV for a gene is then based on which bucket the floating-point UAV value falls into. We then set the start time for this surveil as the maximum of the available time for the target and available time for the UAV (Algorithm 1, line8). Next, we calculate a tentative surveil time, which is the minimum of ***target.surveil_time*** and 24 hours – start time. This will ensure we only receive credit for the surveil up to 24 hours, the length of our analysis (this problem is set up for daily surveillance schedules) (Algorithm 1, line 9). We then calculate the energy consumed for this surveil as:

$$\min(\text{surveil\_time} * \text{uav.rate\_of\_consumption}, \text{uav.max\_energy} - \text{uav.consumed\_energy})$$

Given this energy value, we calculate what percentage of the surveil completed and multiply that by the previously calculated tentative surveil time (Algorithm 1, lines 12-14). This provides the final surveil time for this gene and ensures we only receive credit when there is time remaining and a UAV has energy available. Finally, we update the available time values for the UAV and the target, and the consumed energy for the UAV (Algorithm 1, lines 15-17). The global energy is calculated after all the genes have been iterated over (Algorithm 1, line 19).

4. RESULTS

For our experiments, the NSGA-II was configured in the following manner. The population size was 100 chromosomes, the initial population is completely random, the probability of crossover is 75%, and the probability of mutation is 1%. Fitness evaluations were performed in parallel. We let the algorithm run for 1 minute before termination. All tests were run on a single node consisting of two 2.8 Ghz Intel Xeon E5-2680 V2 processors with 128 GB of RAM, where each processor has 20 threads.

Figure 7, below, shows the results of our experiment. The larger plot in the lower right side of the figure presents the Pareto front between our two objectives, total energy consumed and surveil value. This curve shows the trade-off that exists between the two. That is, as the UAVs consume more energy, they also earn more value. From the front, we see there is an inflection point around 2.5 units of energy, where before this point, we see large increases in value per unit of energy spent, after this point the value earned per unit of energy diminishes with increasing energy. This analysis can be used to inform decision makers on the correct policies for their environments.

Algorithm 1 Fitness Evaluation

Input: chromosome C, $numUavs$, $numTargets$

Output: value, total_energy

1: $uav\_times[numUavs] = 0$
2: $uav\_energy[numUavs] = 0$
3: $target\_times[numTargets] = 0$
4: $value = 0$
5. **for** $gene \in sorted(C.genes)$ **do**
6: $u = gene.uav$
7: $t = target\_uav$
8. $start\_t = \max(target\_times[t], uav\_times[u])$
9: $survey\_t = \min(t.survey\_time, 24\text{-}start\_time)$
10: $exp\_energy = \min(u.energy\_rate * survey\_t)$
11: $energy = \min(exp\_energy, u.max.energy - uav.energy[u])$
12: $survey\_t = survey\_t * (energy / exp\_energy)$
13: $percent\_t = (survey\_t / t.survey\_time)$
14: $value+ = u.value * t.priority * percent\_t$
15: $target\_times[t] = start\_t + survey\_t$
16: $uav\_times[u] = start\_t + survey\_t$
17: $uav\_energy[u]+ = energy$
18: **end for**
19: $total\_energy = \Sigma(uav\_energy)$

Figure 6: Pseudo code for energy calculation

Ideally, Pareto fronts should have solutions as evenly distributed as possible along the entire length of the front. We see that below the inflection point, there are less solutions than above. While we plan to explore this more in future work, we suspect part of the reason this occurs is due to a non-uniform distribution of solutions in the search space [15]. That is, for the smaller energy solutions, there are fewer unique representations of non-dominated solutions than for higher energy solutions. This means that even though two chromosomes may have completely different representations in the solution space, their fitness evaluates to the same values in the objective space.

The smaller plots that surround the Pareto front illustrate the actual UAV-to-target allocations for five solutions along the front. Unsurprisingly, the lower energy solutions use fewer UAVs and surveil less targets, resulting in less value earned. As we move along the front to higher energy consuming solutions, both the number of UAVs used and the targets surveilled increase, resulting in more value earned. From these plots, we can extract some interesting observations. For example, one can quickly see which targets are more profitable to surveil; given our environment, we see that target 9 is never surveilled indicating the value-to-energy spent ratio is too high given the energy and time constraints. We also see that targets 3 and 6 are the most profitable to surveil. In future work, we can use these observations to intelligently seed our GA with initial solutions to aid in the search process [16]. Furthermore, decision makers could use observations like these and potentially adjust the priority value of a given target or procure additional UAVs to maximize the efficiency of their overall missions.

5. Related Work

Many interesting problems consider multiple objectives, but multi-objective optimization is challenging because there typically does not exist a single superior solution for all objectives. Instead, there is a set of superior solutions that are non-dominated and form the Pareto Front [17]. Calculating Pareto fronts can be computationally expensive due to solving numerous variations of the optimization problem. Scalarization techniques are often used to convert multi-objective problems in to a set of scalar optimization problems. Common scalarization approaches include the hybrid and elastic constraint methods [18], Benson's Algorithm [19, 20], and Pascoletti-Serafini scalarization [21]. Approaches such as weighted sums, ε-constraint, and normal boundary intersection are generalized by Pascoletti-Serafini scalarzation.

In addition to scalarization methods, there exist numerous evolutionary approaches for finding Pareto Fronts. This includes multi-objective ant colony optimization [22], multi-objective GA (MOGA) [23], strength Pareto evolutionary algorithm (SPEA2) [24], and the Pareto archived evolution strategy (PAES) [25]. One benefit of evolutionary algorithms over traditional scalarization methods is that they directly provide the set of solutions the make up the Pareto front, instead of optimizing for a single solution at a time. In this work, we use the NSGA-II [14, 15] as our multi-objective algorithm (due to our prior experience and success in using it for multi-objective resource allocation problems [26, 27, 28], but it would be feasible to use any of the above approaches to achieve the same analysis.

6. Conclusions and Discussion

Here we described algorithms, performance measures, and resource allocations required to manage and successfully plan a set of regularly scheduled surveillance requests. The results illustrated the benefits of the algorithms for making allocation decisions in a heterogeneous, oversubscribed, real-time planning environment.

This paper is initial research needed to form a basis for future dynamic environment work, where conditions change, as situational awareness information is obtained, the surveillance schedule could be re-computed [28]. Each mission task will have a time-varying weight (importance) that will be utilized to measure success/failure of the planning algorithms [3].

The preliminary results are encouraging and can form a basis for more research and analysis. For future papers, we intent to open the problem to more constraints to include distance from target and weather conditions that affect the value of a given surveil.

References

1. J. Crowder and J. Carbone J, "Autonomous mission planner and supervisor (AMPS) for UAVs," *20th International Conference on Artificial Intelligence*, July 2018.
2. S. Ali, H. Siegel, M. Maheswaran, D. Hensgen, and S. Ali, "Representing task and machine heterogeneities for heterogeneous computing systems," *Tamkang Journal of Science and Engineering, Special Issue, Invited*, vol. 3, no. 4, pp. 19-207, Nov. 2000.
3. E. Yeonju and H. Bang, "Cooperative task assignment/path planning of multiple unmanned aerial vehicles using a genetic algorithm," *Journal of Aircraft*, vol. 46, no. 1, pp. 338-343, Jan. 2009.
4. S. Mittal and K. Deb, "Three-dimensional offline path planning for UAVs using multi-objective evolutionary algorithms," *2007 IEEE Congress on Evolutionary Computation*, pp. 3195-3202. doi: 10.1109/ CEC.2007.4424880, Sep. 2007.
5. I. Nikolos, K. Valavanis, N. Tsourveloudis, and A. Kostaras, "Evolutionary algorithm based offline/online path planner for UAV navigation," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 33, no. 6, pp. 898-912, doi: 10.1109/ TSMCB.2002.804370, Dec. 2003.
6. G. li, G. Zhou, J. Yin, and Y. Xiao, "A UAV scheduling and planning method for post-disaster survey," *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences,* vol. XL-2, pp. 169-172, Nov. 2014.
7. C. Tunc, D. Machovec, N. Kumbhare, A. Akoglu, S. Hariri, B. Khemka, and H. J. Siegel, "Value of service-based resource management for large-scale computing systems," *Cluster Computing*, vol. 20, issue 3, pp. 2013-2030, Mar. 2017.
8. J. Crowder and J. Carbone, "An agent-based design for distributed artificial intelligence," *International Conference on Artificial Intelligence*, July 2017.
9. J. Holland, *Adaptation in Natural and Artificial Systems*Ann Arbor, MI: The University of Michigan Press, 1975.
10. L. Davis (Ed.), *Handbook of Genetic Algorithms*, Van Nostrand-Reinhold, New York, 1992.
11. E. Hou, N. Ansari, and H. Ren, "Genetic algorithms for multiprocessor scheduling," *IEEE Transactions on Parallel Distributed Systems*, vol. 5, no. 2, pp. 113-130, Oct. 1991.
12. J. Ribeiro Filbo and P. Treleaven, "Genetic algorithms programming environments," *IEEE Computer*, vol. 27, no. 6, pp. 28-43, June 1994.
13. M. Srinivas and L. Patnaik, "Genetic algorithms: A survey," *IEEE Computer*, vol. 27, no. 6, pp. 17-26, June 1994.
14. K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, Apr. 2002.
15. F. Domingo-Perez, J. Lazaro-Galilea, A. Wieser, E. Martin-Gorostiza, D. Salido-Monzu, and A. Llana, "Sensor placement determination for range-difference positioning using evolutionary multi-objective optimization," *Expert System with Applications*, vol. 47, pp. 95-105, Apr. 2015.
16. D. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley Publishing, Inc., Reading, MA, 1989.
17. V. Pareto, *Cours d'Economie Politique*, Lausanne: F. Rouge, 1896.
18. M. Ehrgott, "Multicriteria Optimization," *Springer-Verlag New York Inc.*, New York, 2005.
19. H. Benson, "An outer approximation algorithm for generating all efficient extreme points in the outcome set of a multiple objective linear programming problem," *Journal of Global Optimization*, vol. 13, no. 1, pp. 1–24, July 1998.
20. A. Lohne, "Vector Optimization with Infimum and Supremum," ser. *Vector Optimization*. Berlin, Heidelberg: Springer Berlin Heidelberg, July 2011.
21. G. Eichfelder, "Adaptive Scalarization Methods in Multiobjective Optimization". Springer, Apr. 2008.
22. S Iredi, D. Merkle, M Middendorf, "Bi-criterion optimization with multi colony ant algorithms," *Evolutionary Multi-Criterion Optimization (EMO 2001*), *Lecture Notes in Computer Science*, vol. 1993, Springer, Mar. 2001.
23. C.M Fonseca and P.J. Flemming, "Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization," *Fifth International Conference on Genetic Algorithms,* pp. 416-423, July 1993.
24. E. Zitzler, M. Laumanns, L. Thiele, "SPEA2: Improving the strength Pareto evolutionary algorithm," *Swiss Federal Institute Technology*, Zurich, Switzerland, Mar. 2001.
25. J. Knowles and D. Corne, "The Pareto archived evolution strategy: A new baseline algorithm for Pareto multi-objective optimization," *1999 Congress on Evolutionary Computation (CEC99)*, pp. 6-9, July 1999.
26. K. M. Tarplee, R. Friese, A. A. Maciejewski, and H. J. Siegel, "Efficient and scalable Pareto front generation for energy and makespan in heterogeneous computing systems," in *Recent Advances in Computational Optimization* (S. Fidanova, ed.), Studies in Computational Intelligence Series, Springer, Apr. 2014
27. R. Friese, B. Khemka, A. A. Maciejewski, H. J. Siegel, G. A. Koenig, S. Powers, M. Hilton, J. Rambharos, G. Okonski, and S. W. Poole, "An analysis framework for investigating the trade-offs between system performance and energy consumption in a heterogeneous computing environment," *22nd IEEE Heterogeneity in Computing Workshop (HCW 2013)*, pp. 19-30, May 2013.
28. R. Friese, "Efficient genetic algorithm encoding for large-scale multi-objective resource allocation," *9th Workshop on Large-Scale Parallel Processing (LSPP'16)*, May 2016.

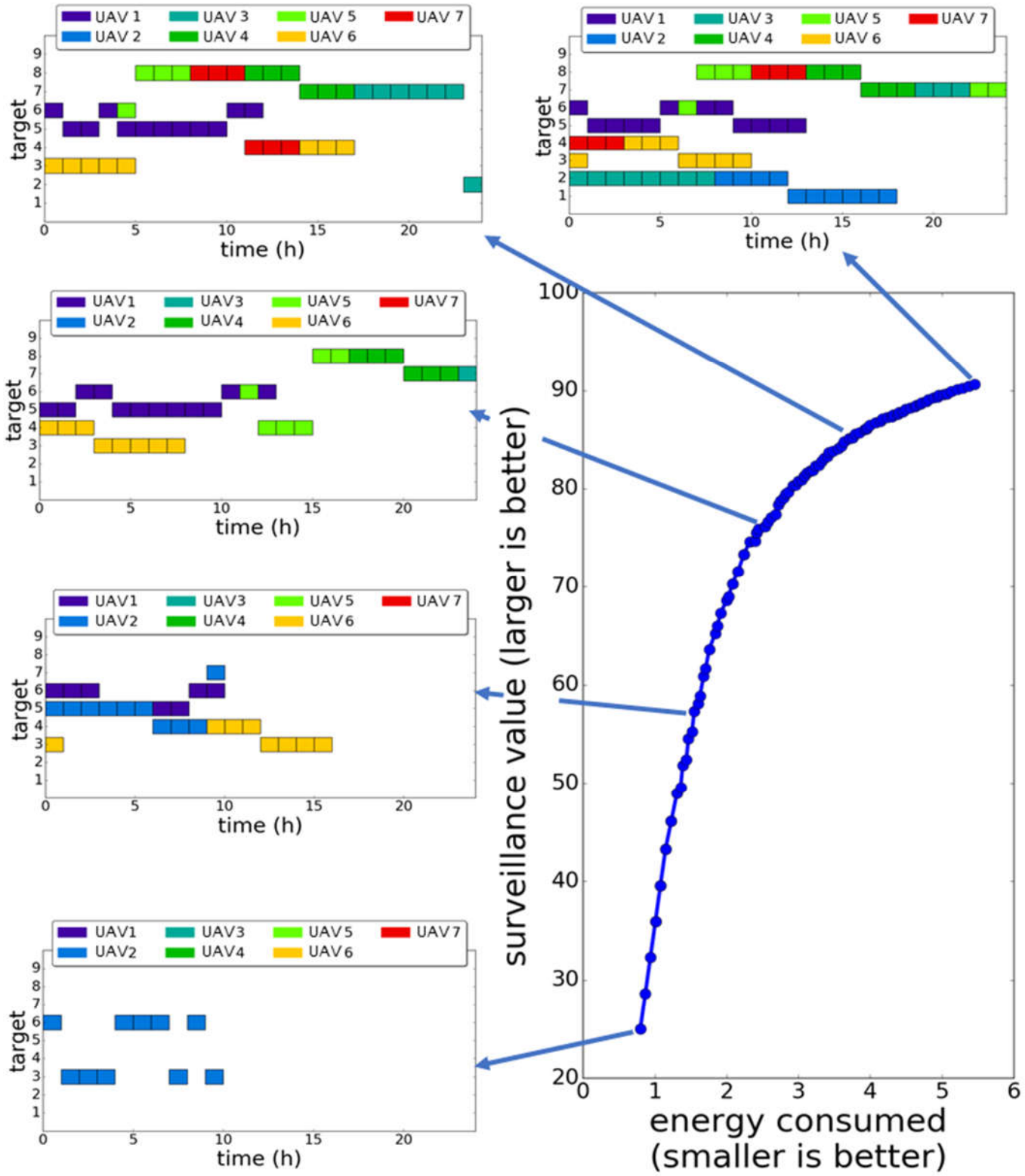


Figure 7: Pareto front (lower right) and individual UAV-to-target allocations for five solutions from the GA run, ranging from least energy (lower left) to most energy (upper right)