

# Adaptive Task Allocation for Search Area Coverage

Ryan J. Meuth<sup>1</sup>, Emad W. Saad<sup>2</sup>, Donald C. Wunsch<sup>3</sup> II, John Vian<sup>4</sup>

<sup>1,3</sup> Applied Computational Intelligence Laboratory at the Missouri University of Science and Technology, Rolla, Missouri,  
e-mail: rmeuth@mst.edu, dwunsch@mst.edu, respectively

<sup>2,4</sup> Boeing Research and Technology, The Boeing Company, Seattle Washington, email: emad.saad@boeing.com,  
john.vian@boeing.com, respectively

**Abstract—** Many operations require an area search area function, including search-and-rescue, surveillance, hazard detection, structures or sites inspection and agricultural spraying. Furthermore, these area search applications often involve varying vehicle and environmental conditions. This paper explores the problem of optimizing the behavior of a swarm of heterogeneous robotic vehicles executing a search area coverage task. Each vehicle is equipped with a sensing apparatus and the swarm must collectively explore an occluded environment to achieve a required probability of detection for each location in the search area. The problem is further complicated with the introduction of dynamic vehicle and environmental properties making adaptability a necessary requirement in order to achieve a high level of mission assurance using unmanned vehicles. Novel methods for search space decomposition and task allocation are presented, with simulated and real-world results utilizing the Boeing Vehicle Swarm Technology Laboratory.

adaptable in real-time in order to maintain the required probability of detection.

These algorithms are uniquely designed to operate in real-time, enabling adaptation to changing vehicle and environmental characteristics, while balancing the task-load between available resources and enabling the seamless control and optimization of swarms of heterogeneous vehicles.

These algorithms have been tested in Boeing's indoor rapid prototyping test-bed that provides a low cost means for technology integration, maturation and demonstration. The test-bed includes over 20 fully autonomous air and ground vehicles and utilizes a motion capture system for indoor localization. The adaptive area coverage task allocation software has been integrated with the defined test-bed Ethernet interface through which it exchanges waypoint commands and vehicle condition and capability data with the vehicles.

## 1. INTRODUCTION

THE search coverage problem for multiple heterogeneous vehicles is unique in that the task is defined in a general way, and must be divided into sub-tasks which are then allocated to each vehicle. The space of these possible assignments is continuous, and increases exponentially with the size and complexity of the region, as well as with the quantity and amount of variation among vehicles, making the problem extremely difficult to solve optimally. As a highly complex problem, multi-vehicle search coverage has been greatly explored in literature [1-3]. Additionally, many of the problem sub-components have been analyzed, including area decomposition, task allocation, and path optimization. The more general problem of collective robotics has also been greatly researched, but continuous environmental effects have rarely been considered [4-6]. In real-world applications such as automated surveillance and search-and-rescue, vehicle and environmental characteristics may be dynamic due to vehicle failures and changing weather conditions, so the optimization methods must be

## 2. BACKGROUND

As a highly complex problem, multi-vehicle search coverage has been greatly explored in literature [1-3, 7-9]. Additionally, many of the problem sub-components have been analyzed, including area decomposition, task allocation, and path optimization. The more general problem of collective robotics has also been greatly researched, but continuous environmental effects have rarely been considered [4-6, 10-12].

### 2.1. Mission Planning Architectures

Several mission planning architectures have been devised, typically consisting of a mission-interpreter, which reads a standardized representation of the mission tasks, a task allocation unit, and a planner. These mission planner architectures, such as GRAMMPS, ALLIANCE, MARTHA, M+, and MURDOCH have rarely handled heterogeneous vehicles, and approach adaptability through continuous re-planning [4, 13-16]. Also, the task of search area coverage is largely unexplored.

### 2.2. Search Area Decomposition

For large continuous tasks, such as search area coverage, it is desirable to decompose the problem space to reduce the complexity of the problem. Several methods have been devised to accomplish this, including grid division methods

of various geometries, and Voronoi divisions [3, 17, 18]. These methods work well for uniform environments and simple sensing apparatuses, but they do not account for the interaction of the vehicle's sensor with the environment.

### 2.3. Task Allocation

Task Allocation is concerned with finding the optimal matching of tasks to vehicles based on vehicle characteristics, task requirements, and a vehicle's current load. Gerkey provides a good overview of several Multi-Robot Task Allocation architectures and analyzes their runtime complexity and optimality [17]. Many of these architectures utilize market based methods where vehicles 'bid' on available tasks, and the vehicle with the lowest bid wins. These methods are distributed and robust, but are often non-optimal, as they are greedy methods operating only on local information [19, 20].

## 3. MISSION PLANNING ARCHITECTURE

The problem of optimal search coverage using multiple heterogeneous vehicles can be decomposed into two coupled problems - that of high-level task allocation for each vehicle, and low-level route optimization for the allocated task. At the highest level, vehicles must be coordinated to ensure that the total search space is covered in minimal time. This coordination takes the form of task allocation for each vehicle in the search space. Here, the task of occluded search-area coverage is explored for multiple heterogeneous vehicles. In scenarios where environments and vehicle capabilities are variable, the task allocation must be adaptable and efficient to handle these changing situations in real-time.

At the lowest level, a single vehicle must plan an optimal path through an allocated search space, which is a sub-region of the total search space. This optimal path is considered as the shortest route through the set of points that allows the vehicle's sensing apparatus to visit all points in the search space.

Given a region to be searched and a vehicle with associated characteristics and capabilities, the task of planning a path through the region is extremely complex if all points in the region are to be considered as part of a possible coverage path for a vehicle. In light of this, a method called Probabilistic Decomposition has been developed for dividing the search space based on the vehicle's sensing characteristics to provide a set of points that, if all points in the set are included in the vehicle's final path, guarantee that complete coverage is achieved, with an efficient set size. This set of points can then be used as an instance of the Traveling Salesmen Problem to find a tour through all points, thus covering the search space.

At this level, the vehicle dynamics, vehicle sensing characteristics, and environmental effects can all be included to affect the optimal solution. With the inclusion of rich

vehicle dynamics, the true value of a path is not only dependant on the distance between its points, but also the vehicle capabilities and characteristics of the vehicle's control system. Each of these elements adds computational complexity to any algorithm, so a new TSP solution method, Clustered Evolutionary Lin-Kernighan or CELK, was developed, but it is not described here. See references [21, 22] for further reading on these algorithms.

In environments and scenarios where vehicle capabilities and the environment may be variable, an arbitrator is used to detect changes in the vehicles and environment, and then decides if a re-plan should be executed.

These four components – decomposition, task allocation, path planning, and arbitrator - are combined into a mission planning architecture shown in Figure 1.

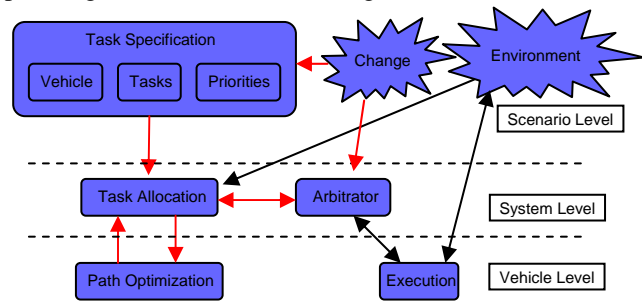


Figure 1 - Adaptive Mission Planner Architecture.

## 4. PROBABILISTIC DECOMPOSITION

In order to decompose the search space, we introduce the probabilistic decomposition algorithm which iteratively divides a plane into a set of non-overlapping regions, or facets, that cover the whole plane, based on the characteristics of the facet vertexes. The subdivision is constructed from a set of 2D points, or facet anchors, that lie at the center of each facet. The borders of each facet lie equidistant from adjacent anchor points. The set of facet vertices are known as the Voronoi diagram of a point set.

The Probabilistic Decomposition algorithm begins by initializing the point set with the vertexes of the bounding region, as well as a point within the bounding region. Beyond this, any further initialization is possible, such as a regular grid, and would serve to reduce the run time complexity of the algorithm. The Voronoi diagram for the point set is then calculated, and the vertexes of each facet are examined, relative to the facet anchor, under a set of constraints. If the facet vertex does not meet the constraint criteria, then the vertex is added as to the anchor point set, adding a new facet, and halving the size of the 3 (or 4, in the case of square facets) adjacent facets along the line between their anchors and the vertex, as shown in Figure 2. This process is repeated for all vertices of all facets until no facets remain unconstrained, and the point set is complete. This method has many desirable properties, most notably that the constraints can be arbitrarily related to the contents of the facet under examination.

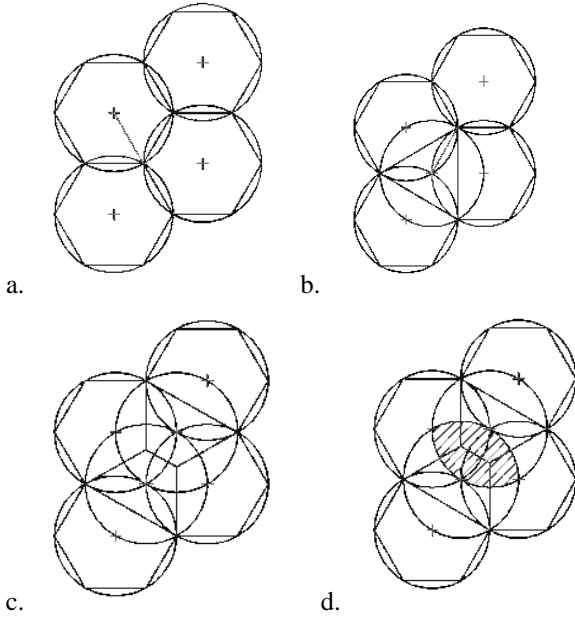


Figure 2 - Constrained Division.

Figure 2 illustrates the constrained division algorithm. Cells are outlined as hexagons, sensor footprints are circular, and anchor points are located at the center of each cell and sensor footprint. 2a. shows the divisions using a simple triangular grid before further constraints are imposed. The gray line in 2a shows a vertex under examination. 2b. shows the division after the vertex from 2a was added as a new anchor point, creating a new cell. 2c. shows the division resulting from an additional anchor point being added. 2d. shows the final division, with the area of highest sensor overlap shaded.

Additional information can be included in the constraint set, such as the probability of observing the true state of points in the facet. Given the probability that an vehicle's sensor will detect the state of a point if it is fully visible, and given the probability of points being visible in the search space (which can correspond to various environmental phenomena, such as vegetation, weather conditions, etc.) a constraint can be constructed that ensures that the vehicle passing through the point set achieves an upper bound on the observability of points in the search space.

The probability of observing the state of a given location is given by Equation 1.

$$P_{obs} = 1 - [1 - (P_{det} P_{vis})]^n \quad (1)$$

$P_{obs}$  represents the probability of observing the state of a location over the entire search sequence, also known as observability.  $P_{det}$  is a value which is related to the sensing capabilities of the vehicle, representing the probability of detecting the true state of a location if it is completely visible during a single pass.  $P_{vis}$  is the probability that a location will be visible. Finally,  $n$  represents the number of times that the vehicle's sensor visits the given location. The probability of observing the state of a given location increases with each

pass of the sensor, so to bound the observability of each point, the appropriate number of passes that result in the observability falling below the desired threshold must be found. This could be accomplished by simply passing an vehicle over the same point repeatedly, or more efficiently, the constrained subdivision method can be used to decrease the size of facets, so that the sensor footprint of an vehicle passing through a facet's anchor overlaps the interior of neighboring facets, increasing the pass count, and increasing the observability of points in that cell without backtracking. In application, a maximum pass number is implemented to prevent large numbers of passes over areas of high occlusion. Cells that meet the maximum pass count but still do not satisfy the observability constraint can be flagged as likely locations for targets.

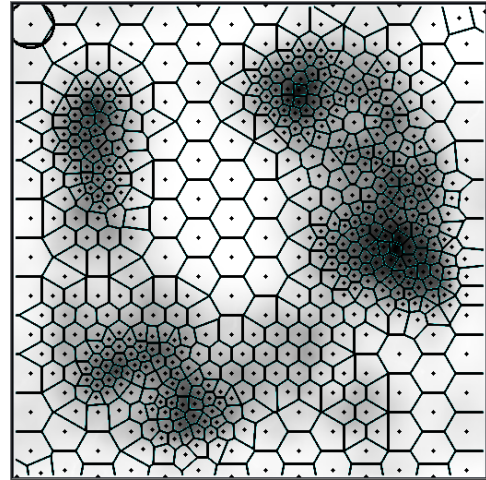


Figure 3 - Triangular Grid Initialized Division under Distance and Observability Constraints.

## 5. EQUILIBRIUM TASK ALLOCATION

The Task Allocator provides the functionality of allocating regions of the search area to each vehicle, according to each vehicle's condition and capabilities. This is accomplished through an iterative process, called Equilibrium Task Allocation, and also through a Hybrid Particle Swarm Optimization method that is applied to high-level assignments. The environmental clusters produced by the Arbitrator are used as initial regions for the search space. An initial assignment is produced by randomly assigning vehicles to the regions. If more than one vehicle is assigned to a region, the region is divided among the vehicles using a gradient-descent based method towards the equalization of the time-loading between all vehicles. Once equilibrium is achieved, vehicles are swapped between regions, and the algorithm repeats until no more swaps are possible, at which point a new random assignment is generated, always keeping the best assignment. In this way, the total area search time minimized, as the amount of time that vehicles are idle is minimized.

The load balancing method uses a gradient-descent based method to equalize the loads between vehicles allocated to a region. Since the relationship between the size of a vehicle's

allocation and a vehicle's loading may not be differentiable, equilibrium error is used, as calculated by finding the deviation of a vehicles' loading from the swarms' average load. The load balancing method is described below:

Load Balance:

- Given a Region and a Set of  $n$  Assigned Vehicles
- Initialize
  - Cluster Region into  $n$  sub-regions
  - Initialize Vehicle subsets with points inside sub-regions
  - Calculate Centroid for each sub-region.
  - Generate a set of weights, one for each vehicle, with initial value  $1/n$ .
  - Add Subsets to Vehicle's Current Point Sets
  - Evaluate to receive a load estimate for each Vehicle
- For 'i' iterations,
  - Calculate Average Load across all vehicles,  $\bar{l}_a$
  - For  $a$  Vehicles,
    - Calculate the Load Error:

$$e_a = \frac{\bar{l}_a - l_a}{\bar{l}_a}$$

- Weight Update:

$$w_a^{i+1} = w_a^i + c(1 - w_a^i)e_a$$

- Normalize Weights to  $|w_a| = 1$
- For each unreserved point in region,
  - Compared weighted distances to each vehicle's centroid.
  - The vehicle with winning centroid reserves that point and adds it to sub-region.
- Evaluate for new load estimate.
- Return Vehicles with Added Subsets.

### 5.1. Hybrid Particle Swarm Task Allocation

A different implementation of task allocation uses a two-stage process to allocate vehicles to search regions. The first stage assigns a vehicle to a region or regions, and the second stage modifies the size of the region(s) in an effort to balance the time-load across all vehicles in the scenario. Currently, the assignment stage uses an exhaustive random selection method to try vehicle-region assignments. This is very inefficient, as the algorithm will attempt allocations until all assignments have been evaluated or until an evaluation limit is reached. It is expected that this stage of the process could benefit from either an evolutionary component or, more appropriately, a particle swarm optimization component.

Particle swarm optimization (PSO) is an evolutionary computation technique developed by Kennedy and Eberhart

[23-26]. Similar to EA, the PSO algorithm is a population based optimization technique in which the system is initialized with a population of random potential solutions and the algorithm searches for optima satisfying some performance index over generations. It is unlike an EA, however, in that each potential solution is also assigned a randomized velocity, and the potential solutions, called *particles*, are then "flown" through the m-dimensional problem space.

$$V_i(t+1) = w \times V_i(t) + c_1 \phi_1 (P_b(t) - X_i(t)) + c_2 \phi_2 (P_g(t) - X_i(t)) \quad (2)$$

Changing the velocity this way enables the particle  $i$  to search around its individual best position  $P_b$  and the global best position  $P_g$ . Based on the updated velocities, each particle changes its position according to Equation 3.

$$X_i(t+1) = X_i(t) + V_i(t+1) \quad (3)$$

The velocity update of a PSO particle given in (5) consists of three parts. The first is the momentum part, preventing abrupt changes. The second part represents learning from its own flying experience. The third part represents the collaboration among particles – learning from the group's best flying experience. The balance among these three parts determines the balance of the global and local search ability and therefore the performance of a PSO [26].

The inertia weight  $w$  controls the balance of global and local search abilities. A large  $w$  facilitates the global search while a small one enhances the local search. The introduction of an inertia weight also frees the selection of maximum velocity  $V_{max}$ . The  $V_{max}$  can be reduced to  $X_{max}$ , the dynamic range of each variable, which is easier to learn; the PSO performance is as good or better [23].

The addition of a particle swarm component to task allocation would take the form of a Binary Particle Swarm Optimization method (BPSO), where each particle represents the allocation of all vehicles to all areas [BPSO Ref]. Each particle will consist of  $m$  by  $n$  bits, where  $m$  is the number of vehicles in the scenario and  $n$  is the number of regions. Using this method, simple bit masks can be applied to filter out 'undesirable' allocations, just as ground-based vehicles being allocated to unreachable areas. In BPSO, the position update function becomes a binary decision, equation 4.

$$x_{id} = \begin{cases} 1 & \text{if } r < s(v_{id}) \\ 0 & \text{if } r > s(v_{id}) \end{cases} \quad (4)$$

Particle swarm optimization has the advantage of quick and detectable convergence on minima, though it may not always converge to the global minimum.

$$s(v_{id}) = \frac{1}{1 + \exp(-v_{id})} \quad \text{and } r \sim U(0,1) \quad (5)$$

The hybrid particle swarm optimization method combines a real-valued and a binary valued particle position under a single fitness value and evaluation. Applied to task allocation in this problem domain, the binary particle position represents the assignment of vehicles to regions in the environment. The real-valued position represents a center point in the search space and the percent of the region allocated to the vehicle.

## 6. HBPSO TASK ALLOCATION RESULTS

The method for estimating the loads of the agents principally determines the value of the Equilibrium Task Allocation method. Estimation methods can range in accuracy from heuristics to high-fidelity simulation of the vehicles. Here two estimation methods are compared – a simple heuristic based on the standard deviation of the points in an agent’s allocation, and a simple path optimization method. The heuristic method first calculates the centroid of an agent’s assigned path points, then sums the travel times between the centroid and every point in the path. This provides a deterministic, precise method for assignment evaluation that has only  $O(n)$  complexity. This heuristic method is compared to a single-pass path optimization, where a random path is generated and optimized to a local minimum. The resulting cost of this path is used as the assignment value estimate. Below, the two assignment estimation methods are compared in Tables 4 and 5. Waste is taken as the amount of time agents idle, having completed their assigned tasks.

Table 4 - Cover Time of Two Agents, Differing in Mobility.

	Path Eval. - No Angle		Path Eval. - Angle		Heuristic	
	Cover Time	Waste	Cover Time	Waste	Cover Time	Waste
Average	2517.28	532.20	2410.85	332.18	2407.35	352.13
Std. Dev.	188.09	269.52	124.36	183.96	47.84	72.24

Table 5 - Cover Time of Two agents, Differing in Sensor Footprint.

	Path Eval. - No Angle		Path Eval. - Angle		Heuristic	
	Cover Time	Waste	Cover Time	Waste	Cover Time	Waste
Average	3490.53	1099.83	2780.50	264.53	2727.10	158.38
Std. Dev.	306.67	415.68	98.61	160.56	57.17	77.16

In both situations, the heuristic method provides consistent results, while at the same time providing good estimates of cover time and minimizing waste. Computationally, the heuristic method is also significantly faster.

## 7. EQUILIBRIUM TASK ALLOCATION EFFICIENCY

The efficiency of the Equilibrium Task allocation method was evaluated in six scenarios, detailed in Table 6. The environment size was held constant in each scenario.

Table 6 - Task Allocation Evaluation Scenarios.

Scenario	Agents	Environment
1	2, Identical	Uniform
2	2, Different	Uniform
3	2, Different	Variant
4	4, Identical	Uniform
5	4, Different	Uniform
6	4, Different	Variant

Table 7 - Divide and Swap Task Allocation Performance.

Scenario	# Agents	Single	Multiple	Speedup	Efficiency
1	2	3598.625	2013	1.787692	0.893846
2	2	3598.625	2694	1.335793	0.667896
3	2	3656	2799	1.306181	0.65309
4	4	3598.625	1072.5	3.355361	0.83884
5	4	3598.625	2005.5	1.794378	0.448594
6	4	3656	2114	1.729423	0.432356

For simple scenarios the efficiency of the Equilibrium Task Allocation algorithm is high, nearly 90% in some cases. In more complex scenarios consisting of heterogeneous agents, the efficiency is necessarily lower. The introduction of variant terrain does not greatly impact the quality of the task allocation, suggesting that the algorithm approaches the maximum efficiency for the scenario.

## 8. HBPSO TASK ALLOCATION EFFICIENCY

The efficiency of Hybrid Binary Particle Swarm Optimization (HBPSO) Task Allocation was evaluated by the same method as Equilibrium Task Allocation, described above. The results are shown in Table 8. The Hybrid Particle Swarm Optimization Task Allocation method performed very well in the simplest scenario, achieving 99% efficiency. However, in all other scenarios the HBPSO method performed worse than EqTA, though not significantly.

Table 8 - HBPSO Task Allocation Performance.

Scenario	# Agents	Single	Multiple	Speedup	Efficiency
1	2	3598.625	1816.5	1.981076	0.990538
2	2	3598.625	2994.5	1.201745	0.600872
3	2	3656	3096	1.180879	0.590439
4	4	3598.625	1167	3.083655	0.770914
5	4	3598.625	2638	1.364149	0.341037
6	4	3656	2172	1.683241	0.42081

In the heterogeneous 2-agent cases, the HBPSO method showed very little difference in efficiency from the uniform to the occluded environment case, suggesting that the HBPSO method was approaching the maximum efficiency in these cases. The differences between the two methods are likely caused by the inherent random nature of the HBPSO method, and its inability to converge on an equilibrium state as fast as EqTA.

## 9. HBPSO vs. EQUILIBRIUM TASK ALLOCATION

The Hybrid Particle Swarm Optimization and Divide and Swap Task Allocation methods were each evaluated in several scenarios. The search area size was kept constant while the number of agents, their characteristics, and the environmental characteristics were varied, providing scenarios of varying complexity. The amount of time required to find an efficient task allocation for each scenario is shown in Figure 13. The HBPSO method is inefficient for simple problem sizes, but has a closer to linear behavior compared to EqTA on more complex problem scenarios. This is likely due to the exhaustive regional assignment search of EqTA, while HBPSO may cull highly inefficient assignments from the search quickly. However, the task allocation quality of HBPSO is inferior to that of EqTA in every scenario, as shown in Figure 14. This is likely due to the slow convergence of HBPSO's real-valued particle positions, while EqTA quickly converges on equilibrium.

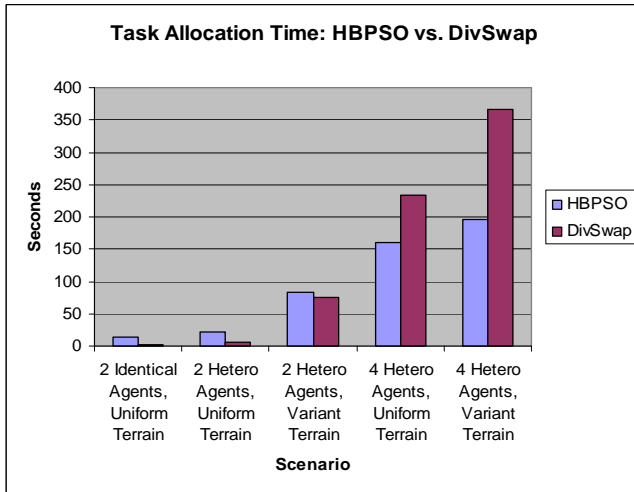


Figure 13 - Time to Complete Task Allocation of HBPSO and EqTA.

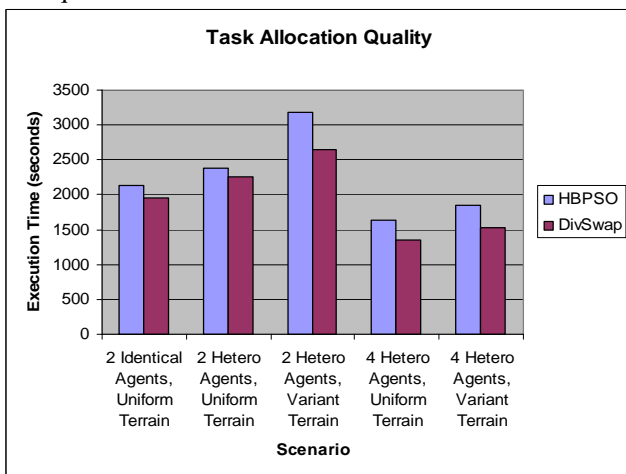


Figure 14 - Task Allocation Quality of HBPSO and EqTA.

## 10. BOEING SWARMS TEST-BED

### 10.1. Vehicle Swarm Technology Lab

The above algorithms have been tested in Boeing's Vehicle Swarm Technology Lab (VSTL) [27] on air and ground vehicles. VSTL is a 100x50x20ft indoor rapid prototyping test-bed that provides a low cost means for technology integration, maturation and demonstration.

*Architecture*—The overall VSTL test-bed architecture is comprised of hardware and software elements. Hardware elements include a high-accuracy, low-latency, vision-based, motion capture position reference system and a number of hardware vehicles under test. Each vehicle under test is equipped with its own on-board controller, health-monitoring, and communication sensor payload. Communication between the various software elements is via either of two data buses. One bus is used for transmitting time critical vehicle position and attitude data. The second bus is used for transmitting health, condition, and capability data as well as vehicle commands. The interaction between these applications is through UDP Ethernet packets. The architecture also supports TCP for less time-critical data when accurate data delivery is required.

The adaptive area coverage task allocation software has been integrated with the defined test-bed dual bus network interface through which it receives the vehicles position and capabilities data and sends waypoint commands waypoint commands to the vehicles.

*Vehicles*—VSTL uses common RC vehicles equipped with Boeing's custom common electronic hardware. The common hardware is a modular electronic board that allows easy integration of new types of vehicles and allows easy interoperability with other test-bed components. The tests below have been conducted using quadrotors as air vehicles and tanks as ground vehicles shown in Figure 16.



Figure 16 – Quadrotor (Left) and Tank (Tight) Test Vehicles Use Boeing Common Hardware.

### 10.2. VSTL Test Results

The following 2 scenarios have been tested with the above algorithms to demonstrate some of the algorithm capabilities. Both tests have been done using the Divide and Swap Task Allocation method. The search area size was 8x6 meters and flight height was 1.5 m.

### 10.3. Two Air Vehicles Test with Different Speeds

The first test included 2 air vehicles with different speeds in order to show the load balancing capability of the Divide and Swap Task allocation. The vehicles sensor radius was set to 0.8 m, and vehicle speed was 0.6 m/s for one and 0.3 m/s for the other.

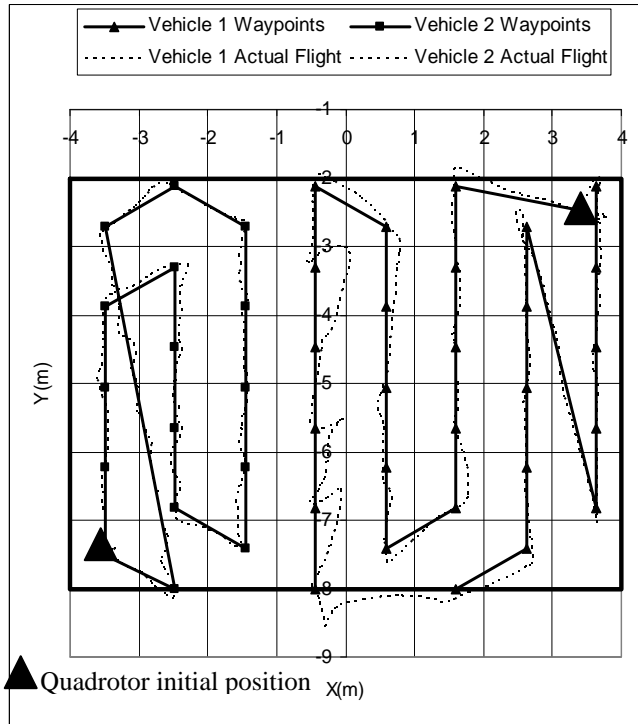


Figure 17 – Planned Waypoints and Actual Paths for Vehicles. The Task Allocation Algorithm Assigned a Longer Path to Vehicle 1 Due to its Higher Speed in order to achieve equilibrium.

Figure 17 shows the flight test results. Vehicles 1 and 2 completed the search in 91.44 sec and 85.75 sec respectively. Thus although vehicle 1 speed is double that of vehicle 2, their search time standard deviation is only 4%. Also notice the deviation of the actual flight path of vehicle 1 from the planned path in segments close to vehicle 2's path due to collision avoidance between vehicles. The vehicle however was still able to achieve its waypoints and guarantee the coverage.

### 10.4. Heterogeneous Vehicles Test

This test was designed to demonstrate advanced features of the adaptive task allocation algorithm by including heterogeneous vehicles, a non uniform occlusion map, and dynamic vehicle capabilities. The test included 2 air vehicles (quadrotors) and 2 ground vehicles (tanks) with different capabilities as shown in Figure 18. The quadrotors sensor radius was 1.2 m, and their speed was 0.6 m/s. The Tank's sensor radius was 0.6 m, and their speed was 0.4 m/s. Max number of passes was set to 3 for all vehicles. At 58.32 seconds from the beginning of the search, quadrotor 1 changed altitude to 0.75 m and sensor radius to 0.6 m which triggered a replan. Figure 19 shows planned and actual

paths as well as the occlusion map. Vehicles have been assigned new paths after the replan which is indicated by the circle for every vehicle. The vehicles adaptively completed the coverage of the whole area despite the sensor radius reduction of quadrotor 1.



Figure 18 – Heterogeneous Area Coverage Test in VSTL. Mission Planner is Shown in Bottom Left.

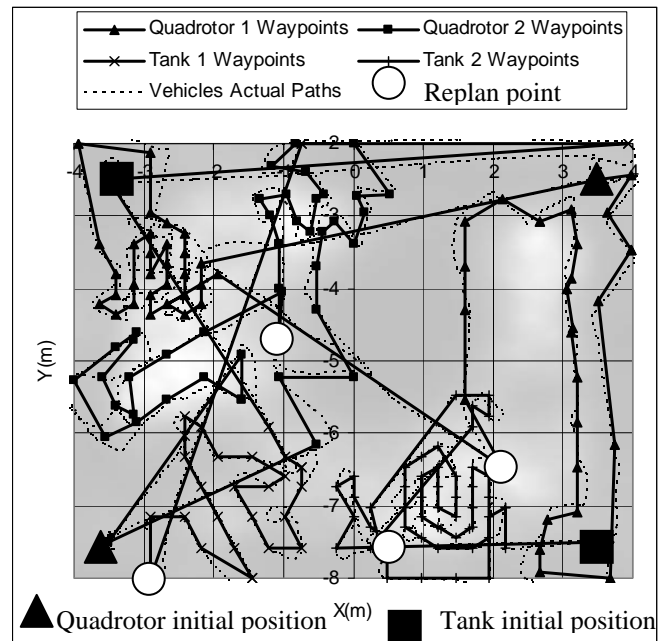


Figure 19 - Planned and Actual Paths and the occlusion map. Brighter regions indicate higher occlusion. The planner waypoints are denser over occluded areas. Quadrotor 1 waypoints are denser after the replan due to its sensor reduction.

## 11. CONCLUSION

This paper presents novel area coverage algorithms that have been validated using Boeing VSTL hardware. Even though the Multi-vehicle Search Area coverage problem is large and complex, several novel methods have been presented that decompose, allocate and optimize the exploration of a search area for multiple heterogeneous vehicles. These new

methods were shown to have good performance and quality, and as they are defined in a general way, these methods are applicable to many other problem domains. The methods have been combined into a mission planner architecture that is able to adaptively control the behavior of multiple vehicles with dynamic vehicle capabilities and environments for mission assurance.

The topic of mission planning architectures and optimization of swarms of autonomous vehicles is a young and exciting field with many opportunities for research. For task allocation, a combination of binary particle swarm optimization and the equal-division algorithm will be explored. Also, more computationally efficient methods for decomposition may be useful, as well as improvements to the clustered evolutionary LK method. In addition to the existing collision avoidance, path deconfliction during planning can improve safety and efficiency.

## REFERENCES

- [1] R. N. De Carvalho, *et al.*, "Complete coverage path planning and guidance for cleaning robots," in *Industrial Electronics, 1997. ISIE '97., Proceedings of the IEEE International Symposium on*, 1997, pp. 677-682 vol.2.
- [2] N. Hazon and G. A. Kaminka, "Redundancy, Efficiency, and Robustness in Multi-Robot Coverage," in *IEEE International Conference on Robotics and Automation*, 2005, pp. 735-741.
- [3] J. S. Oh, *et al.*, "Complete Coverage navigation of cleaning robots using triangular-cell-based map," *IEEE Transactions on Industrial Electronics*, vol. 51, pp. 718-726, June 2004.
- [4] R. Alami, "A General Framework for Multi-Robot Cooperation and its implementation on a Set of Three Hilare Robots," *Experimental Robotics IV*, 1995.
- [5] I. Dutta, *et al.*, "Collective Robotics - a survey of control and communication techniques," in *International Conference on Intelligent Mechatronics and Automation*, 2004, pp. 505-510.
- [6] A. Farinelli, *et al.*, "An analysis of coordination in Multi-Robot Systems," in *IEEE International Conference on Systems, Man, and Cybernetics*, 2003, pp. 1487-1492.
- [7] E. U. Acar, *et al.*, "Sensor-based coverage with extended range detectors," *Robotics, IEEE Transactions on [see also Robotics and Automation, IEEE Transactions on]*, vol. 22, pp. 189-198, 2006.
- [8] D. I. Latimer, *et al.*, "Towards sensor based coverage with robot teams," in *Robotics and Automation, 2002. Proceedings. ICRA '02. IEEE International Conference on*, 2002, pp. 961-967 vol.1.
- [9] M. Moors, *et al.*, "A Probabilistic Approach to Coordinated Multi-Robot Indoor Surveillance," Department of Computer Science III, University of Bonn, Germany.
- [10] H. Li, *et al.*, "An optimization algorithm for the coordinated hybrid agent framework," in *IEEE Conference on Systems, Man, and Cybernetics*, 2005, pp. 1730-1735.
- [11] L. Lin and Z. Zheng, "Combinatorial Bids based Multi-Robot Task Allocation Method," in *IEEE International Conference on Robotics and Automation*, 2005, pp. 1145-1150.
- [12] N. Zhang and D. C. Wunsch, II, "A comparison of dual heuristic programming (DHP) and neural network based stochastic optimization approach on collective robotic search problem," in *International Joint Conference on Neural Networks*, 2003, pp. 248-253.
- [13] S. Botelho and R. Alami, "M+: a scheme for multi-robot cooperation through negotiated task allocation and achievement"," in *IEEE Intl. Conference on Robotics and Automation*, Detroit, MI, 1999, pp. 1234-1239.
- [14] B. L. Brumitt and A. Stentz, "GRAMMPS: A Generalized Mission Planner for Multiple Mobile Robots in Unstructured Environments," in *IEEE International Conference on Robotics and Automation*, 1998.
- [15] B. P. Gerkey and M. J. Mataric, "Sold!: Auction methods for multi-robot coordination," *IEEE Transactions on Robotics and Automation*, vol. 18, pp. 758-768, 2002.
- [16] L. E. Parker, "L-ALLIANCE: A mechanism for Adaptive Action Selection in Heterogeneous Multi-Robot Teams," Oak Ridge National Laboratory October 1995.
- [17] B. P. Gerkey and M. J. Mataric, "Multi-robot task allocation: Analyzing the complexity and optimality of key architectures," in *IEEE International Conference on Robotics and Automation*, 2003, pp. 3862 - 3868.
- [18] K. Nagatani and H. Choset, "Toward robust sensor based exploration by constructing reduced generalized Voronoi graph," in *Intelligent Robots and Systems*, 1999, pp. 1687 - 1692.
- [19] H. Hanna, "Decentralized approach for multi-robot task allocation problem with uncertain task execution," in *International Conference on Intelligent Robots and Systems*, 2005, pp. 535-540.
- [20] E. H. Ostergaard, *et al.*, "Multi-Robot task Allocation in the Light of Uncertainty," in *IEEE International Conference on Robotics and Automation*, 2002, pp. 3002-3007.
- [21] R. J. Meuth, "Adaptive Multi-Vehicle Mission Planning for Search Area Coverage," Missouri University of Science and Technology, Master's Thesis, 2007.
- [22] R. J. Meuth and D. C. Wunsch, II, "Divide and Conquer Evolutionary Tsp Solution for Vehicle Path Planning," presented at the Congress on Evolutionary Computation (WCCI'08), 2008.
- [23] R. Eberhart and Y. Shi, "Particle Swarm Optimization: Developments, Applications, and Resources," in *Conference on Evolutionary Computation*, 2001, pp. 81-86.
- [24] J. Kennedy and R. Eberhart, "Particle Swarm Optimization," in *IEEE International Conference on Neural Networks*, 1995, pp. 1942-1948.
- [25] J. Kennedy, *et al.*, *Swarm Intelligence*. San Mateo: Morgan Kaufmann, 2001.
- [26] Y. Shi and R. Eberhart, "Parameter Selection in Particle Swarm Optimization," in *Conference on Evolutionary Computation*, 1998.
- [27] E. Saad, J. Vian, G. Clark, and S. Bieniawski, "Vehicle Swarm Rapid Prototyping Test-bed", in *Proc. AIAA Infotech@Aerospace Conference and Exhibit*, Seattle, WA, 2009.