ARCHIMEDES: Active Reasoning Conducted with Heterogeneous Information Monitors for Evading Dangers and Extended Surveillance

Simon Schoenbeck
SIFT, LLC
Minneapolis MN, USA
sschoenbeck2@unl.edu

Ugur Kuter
SIFT, LLC
Minneapolis MN, USA
ukuter@sift.net

Abstract

ARCHIMEDES is a suite of algorithms for planning and task allocation to monitor and surveil a geometrically-defined area with a group of heterogenous agents. We focus on maximizing the information gained over a fleet of such agents. The multi-step process allows for our force-based system to find a solution faster by generating reasonable starting points for agents. We present our proposed algorithms, compare them to the most relevant related work via a walkthrough discussion, and describe experimental results from ARCHIMEDES in many mathematically-abstract scenarios.

1 Introduction

Unmanned aerial vehicles (UAVs), and robotic agents in general, have become common in information-gathering roles in recent years. In these situations, maximizing the information gained over a fleet of UAVs is the desired outcome. This is accomplished by positioning available UAVs so that their sensors can gain information from points of interest (POI). These positions are limited since UAVs do not have infinite flight range and must avoid objects in the environment.

Current approaches for multi-agent multi-task assignment and similar problems rely on symbolic constraints and discrete assignments (Robinson, Su, and Zhang 2021; Capezzuto, Tarapore, and Ramchurn 2020). Finding the optimal solution for assignments is NP-Hard so estimating techniques have been developed (Gerkey and Matarić 2004; Aziz et al. 2021). These algorithms find approximate solutions which are useful for determinining which agents should be assigned to a certian task but the nature of these algorithms limit their scalability and application to assigning positions to UAVs. The total number of possible positions can be significantly larger than the number of tasks or agents depending on the size and detail of the modeled environment. In many tasking applications physical position is not considered, in our model the physical position is used to determine if an agent is able to collect information from a POI.

This complexity is compounded by the addition of No-Go Zones (NGZs). Modeling these constraints with existing algorithms is difficult as it is hard to symbolically express how a NGZ impacts a task. A simple model may limit a UAV from being assigned to observing a POI, even if the agent may be able stay outside of a NGZ while observing that POI that is within the NGZ. Another complexity is that heterogeneous UAVs may have different sensors allowing them to gain information from certain POI. Although there exist algorithms for team building and multi-agent task assignment for heterogeneous agents (Czatnecki and Dutta 2019), they consider only attainment goals and time required to achieve those goals, which is not sufficient for information-gain problems. Our system also allows for heterogeneous agents. Agents with different sensors may gather different information. This is similar to coalition (Capezzuto, Tarapore, and Ramchurn 2020) and team building (Georgara, Rodríguez-Aguilar, and Sierra 2021) though this algorithm is meant for the maximization of information gain at a certain time, not the completion of tasks over time.

We propose a three-stage system to position UAVs to maximize information potential gain. The Grid Placement algorithm constructs a set if points that meet all of the geometric constraints, points must be within the defined polygon and not within any NGZs. The Agent-Location Pairing algorithm is an extension of the Gale-Shapely algorithm (Gale and Shapley 1962) and assigns initial positions for agents. These positions are used as inputs for the Information Force System which modifies the position of agents by modeling information as attractive and repulsive forces. We present experimental results showing the effectiveness of the combined functionality provided by our algorithms.

2 Problem Formulation

A limitation of other Multi-Robot Task Allocation (MRTA) (Gerkey and Matarić 2004) approaches is the format of the output. The desired output for our problem is the set of agents and their positions for maximizing information gain. Considering every possible point a task and pairing a finite set of agents to this infinite set of tasks with a brute force solution would take an infinite amount of time. A better solution is necessary.

Our formulation models the locations agents can occupy, the potential information, and the observation ability of the agents. The valid locations for agents are constrained to a polygonal region and not within circular no-go-zones (NGZs). The potential information is expressed as areas of interest or points of interest (POI). The available agents are modeled with agent objects with observational radius and channels. These channels are similar to the compentencies in (Georgara, Rodríguez-Aguilar, and Sierra 2021). Figure 1 shows an example region with agents distributed to observe all of the POI.

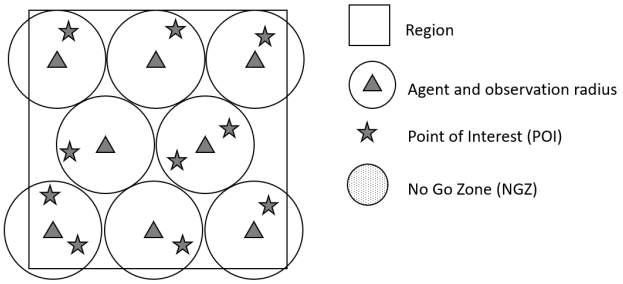

Figure 1: Area of Cover Example

The region in which agents may occupy is modeled as a polygon. This region limits the possible spaces from an infinite 2D plate to a finite area. In the real-world terrain features limit where agents may be positioned within the region. These additional limitations are modeled with the NGZs. The subtraction of the NGZs from the region is the set of all possible places a UAV could validly occupy.

The locations where information is available either as areas or single points are modeled with POI. These POI have a potential information value. Our implementation considers the potential information in the region. The potential information is the difference of the total information available from the POI and the information gathered by the agents by observing the POI. If a POI is unobserved then the potential information near the POI is high. If the POI is being directly observed by an agent, then the potential information is low. A POI that is not directly observed by any agents, but has an agent nearby has a moderate potential information. By modeling the information as a region with differing values for potential information, positioning agents to minimize the potential information maximizes the information gained by the agents. This work is similar to MRTA which maximize the utility by pairing agents to tasks based on the abilities of the agents and the requirements of the tasks.

Our agents have observational ranges and channels. The range value determines how far away an agent can be from a POI and gain information. The channel value must match the channel value of the POI to gain information. These values model the sensor properties. Figure 2 A shows the optimal positioning of an agent between two POI such that it can observe both simultaneously. This positioning is more efficient than assigning an agent to each of the POIs. Figure 2 B shows the desired property of agents with different channels not interfering with each other. Since the agents have different channels and gain information from different POI there is no benefit in increasing the spacing between the agents.

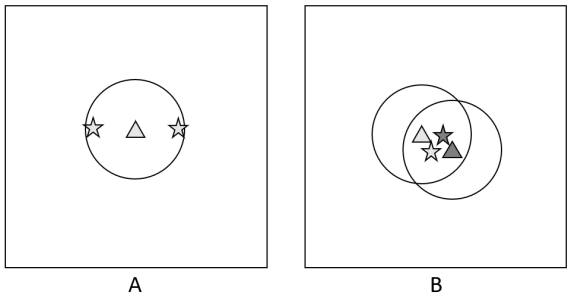

Figure 2: Agent-POI Interaction Example

If we assume all POI are within the region and agents can only be positioned directly over POI this becomes the same setup as a normal MRTA problem. The second assumption causes issues when the POI is within the region, but also within an NGZ. Other approaches could include constraints where a specific agent may not be assigned to a specific task, but this may not include enough detail to describe where a UAV could be positioned outside of a NGZ and still observe a POI within the NGZ Figure 3.

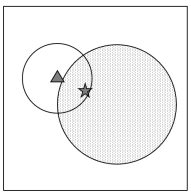

Figure 3: Positioning Example

3 Approach

3.1 System Overview

Our approach improves on existing work by expanding the detail of the model without incurring the computational cost of symbolic constraints. We propose a procedure which contains three distinct algorithms: Grid Placement, Agent-Location Pairing, and Information Force System. Our solution has the Grid Placement algorithm reduce the initial infinite set of locations to a finite set. The Agent-Location Pairing algorithm places agents on these points based on the estimated information gain. These positions are then adjusted by the Information Force System to maximize information gain by applying attractive forces between from POI on agents and repulsive between agents.

If given a random distribution of agents into the region, the Information Force System alone could find a local maximum for the information gain, but the number of time steps required could be large. The Agent-Location Pairing algorithm improves this by positioning agents near POI and only placing necessary agents, therefore reducing the number of time steps needed compared to a random distribution

if agents. The Grid Placement algorithm generates the possible locations for agents for the Agent-Location Pairing algorithm from the region and NGZs. The inputs for the system are: A region R, defined as a list of n points

$$[(x_1, y_1), (x_2, y_2), \dots (x_n, y_n)]$$

A set Z, of size z circular NGZs defined by a center point and a radius

$$\{(x_1, y_1, r_1), (x_2, y_2, r_2), \dots, (x_m, y_m, r_m)\}$$

A set P, of p POIs each defined by an (x, y) coordinate and a association list of k channels c and associated strengths s.

$$(x, y, [(c_1, s_1), (c_2, s_2), ...(c_k, s_k)])$$

A set, A of a available UAVs each defined by an (x, y) coordinate and a association list of k channels c and associated observation radius r.

$$(x, y, [(c_1, r_1), (c_2, r_2), ...(c_k, r_k)])$$

The output is a set U, of u UAVs with positions.

$$[(a_1, (x_1, y_1)), (a_2, (x_2, y_2)), ...(a_u, (x_u, y_u))]$$

3.2 Grid Placement

The Grid Placement algorithm creates a uniform grid of locations such that no point is outside of the given region an no point is within an NGZ. Our algorithm can generate a square grid (SQR) or a hexagonal grid (HEX) pattern Figure 4. The input for the algorithm is the region, the NGZs, the desired number of points, and the grid pattern. This algorithm will not necessarily return the exact number of points as desired. Computing the exact area of the input region, which may be a convex polygon and then subtracting the area of the intersecting circular NGZs is costly. It is simpler to estimate the area of the region and then divide this area by the number of agents to get an estimated area per agent. The square root of this area gives the side length of a square. Overlaying a grid with this spacing on the region and placing produces the SQR grid.

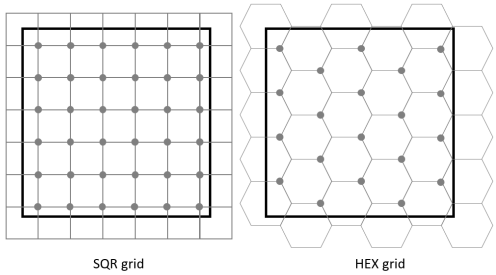

Figure 4: Example of SQR and HEX grid placement

3.3 Agent-Location Pairing

The Agent-Location Pairing algorithm, like many other algorithms for solving MRTA is an expansion of the Gale-Shapely algorithm. Given the available agents, the set of locations from Grid Placement, and the set of POIs the algorithm returns a set of agent-location pairs. The utility of each agent type is calculated for each location based on the information gathering radius of the agents and the nearby POI. This utility function returns the sum of the information gained by placing a certain agent type at a location for each POI. If the POI is within the observation radius the full information is collected. If the POI is outside of the observation radius, but within the influence radius then a fraction of the information is collected, otherwise 0 information is collected. The values for the vision radius and the influence radius are the same values used in the Information Force System.

Data: Agents A, Locations L
Result: V a list of Agent-Location-Value tuples
for $a \in A$ **do**
  for $l \in L$ **do**
    $v \leftarrow 0$;
    if *Channel*(a) =*Channel*(l) **then**
      if $d <=$ *ObservationRadius*(a) **then**
        $v \leftarrow$ AvailableInformation(l);
      else
        if $d <=$ *InfluenceRadius*(a) **then**
          $v \leftarrow$ AvailableInformation$(l)\times$ InfluenceFunction(a, d);
        end
      end
    end
    $V \leftarrow (a, l, v)$;
  end
end

Algorithm 1: Agent-Location Pair Value

While calculating this heuristic a bucketing approach is used to reduce the amount of sorting required. If a minimum utility threshold is set, do not store any pairs that are below the threshold since they are not considered viable pairs. This reduces memory cost and sorting time. Beginning with the bucket with the largest value, the bucket is sorted. Iterate through the agent-location pairs. If the agent has not been assigned to a location and the location has not been assigned more than the threshold number of agents, add the agent-location pair to the list of selected pairs and update the counters for the number of assigned agents and the number of agents assigned to the location. Continue until the bucket is empty then move the to next bucket. Prior to sorting, drop all pairs from the bucket where an agent or location has been already used more than the threshold. This strategy reduces the total sorting time by reducing the total number of elements in the bucket. If the counter for assigned agents equals the number of total agents, then exit the function early. Otherwise continue until the list of pairs is empty. Not all agents may be assigned to a location depending on the utility threshold, the number and distribution of POIs, and the number of agents. The output is an approximation positioning for the agents for the initial agent-location pairs for the Information Force System.

3.4 Information Force System

The Information Force System adjusts the positions of agents within the region by considering the agents as particles that are affected by forces. Agents are attracted by nearby POI based on the value calculated in Algorithm 2. These radii are shown in Figure 5. Agents are repelled by other agents within similar observation properties. This provides small adjustments to the distribution of agents.

It is assumed that all agents are within the region and not within any NGZs. This property is guaranteed by the Grid Placement system. Following this assumption provides a safe fail condition for when a valid new position cannot be found for an agent. Not moving an agent is considered safe since it's current position must be valid. During each time step the new position of each agent is calculated. For each agent their initial velocity is zero. This velocity is summed with the attractive force of each nearby POI and the repellant force of each nearby agent. This velocity vector is summed with the current agent position to calculate the new position. This new position is checked against the region and the NGZs. If the new position is valid then the agent is moved to the new position for the next time step. If the new position is invalid the current agent position will be maintained. This repeats until there is no significant change in agent position of the maximum number of time steps is reached.

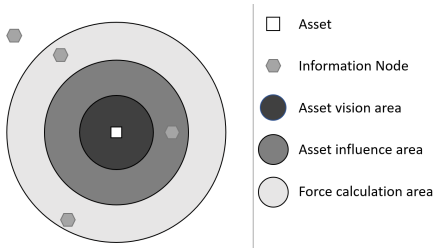

Figure 5: The Important Distances for Computing Forces

3.5 Combined System Advantages

The Information Force System alone is a useful algorithm for spacing agents in a region to maximize information gain. A major limitation of this algorithm is that the final output is dependent on the starting conditions. With random starts it is difficult to estimate the number of required agents without running multiple iterations. These iterations can be costly with high numbers of POI and agents. The Grid Placement and Agent-Location Pairing reduce this computational cost by limiting the number of starting locations and selecting an appropriate number and distribution of agents.

4 Experiments

We conduced experiments on each of the algorithms to compare performance measures. For our experiments we tested our algorithms for scalability by using large numbers of agents and locations. All experiments were conducted on a 12th Gwen Intel® Core™ i7-12700H with 16GB of 4800 MHz RAM. For the Grid Placement algorithm we compared the required loops for the SQR and HEX grid placement algorithms. Our implementation of the HEX algorithm is more complex than the SQR algorithm, but if the HEX algorithm is better at estimating and placing locations then it may require fewer loops. The Agent-Location Paring algorithm is an expanison of the Gale-Shapely algorithm. We tested the original algorithm to our more general algorithm and strategies for improving the runtime. For the Information Force System we tested the improvement in information gain after a set number of iterations. These tests provide insite into properties of the algorithms and the problem space.

4.1 Grid Placement

The Grid Placement algorithm is designed to quickly generate a large number of points that meet all of the geometric constraints. To satisfy the requirement for returning at least as many points as requested, the implementation requires a while loop to rerun the algorithm with tighter spacing. The two grid patterns square grid (SQR) and the hexagonal grid (HEX) were compared on identical test sets to compare the number of loops required to generate the requested number of points. In this experiment 100,000 instances of a random polygon were generated as a region and 0 to 4 NGZs were placed. The random polygon was defined by 3 to 13 points in the range (0,0) to (100,100). For each region and NGZs the grid placement algorithms were requested to generate some number of points. This number was a power of 2 from 1 to 2048.

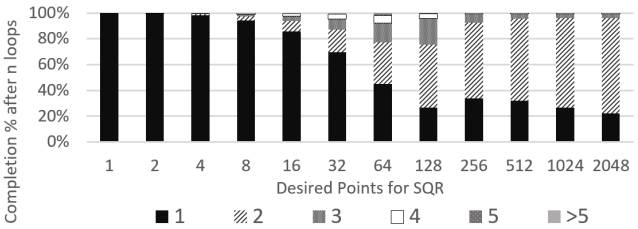

Figure 6: Required Loops for Placing Agents with HEX

The number of times the algorithm needs to loop to complete an instance with the HEX function is shown in Figure 6. More than 88% of the time HEX was able to generate enough points in the first loop. For placing 16 or 32 points the HEX function completed 75.1% and 65.6% respectively, of the instances in one loop.

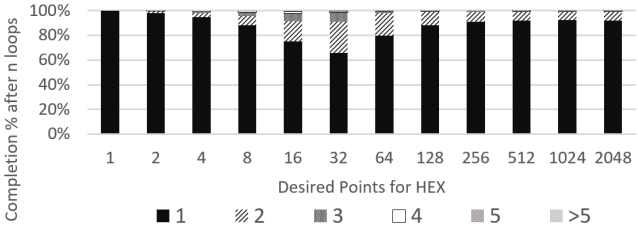

Figure 7: Required Loops for Placing Agents with SQR

The number of times the algorithm needs to loop to com-

plete an instance with the SQR function is shown in Figure 7. More than 61% of the time SQR was able to generate enough points in the first loop. The SQR function required 3 or more loops to complete 22.8% and 24.6% for placing 64 or 128 points respectively.

For each test input the difference of the number of loops taken by the SQR and HEX functions was calcualted. The functions require the same number of loops 57.9%, SQR requires fewer loops 6.4%, and HEX requires fewer loops in 35.7% of instances. We hypothesize that the increased packing density of the HEX grid is a contributing factor to why it completes more instances with fewer loops.

On the same set of test runs we measured the ratio of the number of returned locations to the number of desired locations, $\frac{|L|}{l}$. exhausted Since the Grid Placement algorithm loops while $|L| < l$, therefore $|L| \geq l$ to exit the loop. Our results in Figure 8 show the ratio is always above one, confirming that the loop condition is functioning. When the number of desired locations is low the ratio is often high. We hypothesize that this is because in the cases where the first point selected is not a valid location, the process of shrinking the spacing to add more locations adds too many locations in a single step. There is a significant trend where the HEX grid generates fewer locations than SQR on average. This could be due to the tighter packing of the locations in HEX compared to SQR, and when the spacing value is shrunk the HEX grid provides a better approximation of the area taken per locations than the SQR grid.

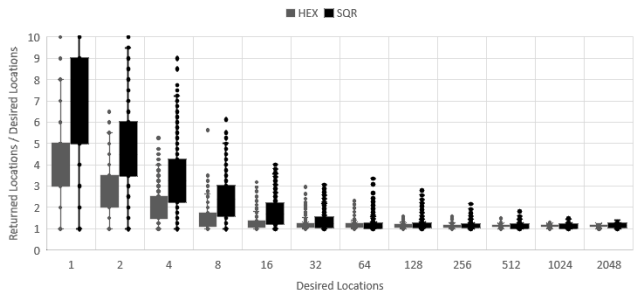

Figure 8: Ratio of Returned Locations to Desired Locations

4.2 Agent-Location Pairing

The original Gale-Shapely algorithm works on the assumptions that there are an identical number of items in the two sets to be paired and that all utility values for the pairing are known (Gale and Shapley 1962). We generalized this algorithm to apply for there to be a different number of agents and location and pairing multiple agents to the same location. This requires the computation and storage of the utility value of every agent-location pair. Fast implementations of the Gale-Shapely algorithm use a data structure where the utility values are sorted. The implementation of the Gale-Shapley algorithm is named MinFrist. Our solution more generalized implementation of the Gale-Shapley is called MinAll. To improve the speed of the algorithm we implemented partitioning and bucketing.

The partitioning solution was developed for improving the speed of calculations that require information about nearby points. For this experiment we considered the utility function as the the distance from an agent's starting point to their paired location and minimizing the total distance traveled. To partition the region we declare a partition size P. The region is then subdivided into squares of side length P. Instead of computing the distance from every agent, to every location, we only compute the distance from an agent to the locations in the same and neighboring partitions. This significantly reduces the number of computations if the agents and locations are evenly distributed. After all of the pairs generated by the partition are exhausted the algorithm then needs to run the MinAll to catch any extra parings that require more than P.

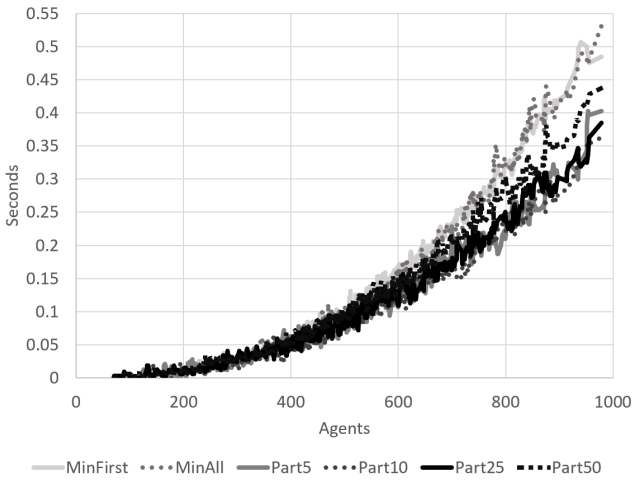

Figure 9: Runtime performance of Partial Sorting

To compare the speeds and accuracy of the algorithms n agents and n locations were generated in a region of size 100 by 100. The algorithms were tasked with creating agent-location pairs while minimizing the sum of distance between paired points. All the algorithms had identical performance for each individual test. This was an expected result since all the algorithms are variations of the same greedy algorithm and this confirms that the property that no optimization is being lost during the partitioning step. Figure 9 shows the average runtime of the algorithms based on the number of agents n. MinFirst and MinAll have nearly the same performance. The Part# algorithms run the partitioning algorithm with partitions of size P. To fully pair all agents MinAll was run on the remaining agents after Part#. Part10 and Part25 were generally the fasted algorithms. We hypothesize that this is because most of the long distance calculations were not done. Since the region was size 100 by 100, most of the distances were calcuated for Part50 but there was still a noticeable speedup over MinFirst and MinAll. This could be due to the filtering of pairs with distances over 50 which reduced the sorting cost.

4.3 Information Force System

The Information Force System is capable of distributing agents without the other algorithms, but is not as efficient. If there are enough information nodes evenly distributed across the region, then agents can be moved to other parts of the region. We generated a random region with 0 to 4 NGZs, 1 to 256 POI, and 1 to 256 Agents randomly scattered within. This input was given 25 and 50 iterations in the Information Force System to generate new positions for the agents. The sum of the potential information was measured at 0, 25, and 50 iterations. For example if there is 1 agent and 1 POI where the agent cannot observe the POI then the initial information collected is 0. If after 50 iterations the agent can observe the POI then the information collected is 1. The Information Gain is the difference between the initial information collected and the information collected after the 50 iterations. The % Information Gain is the Information Gain divided by the Total Information from all the POI. We cannot expect perfect information gain since it it possible for a POI to be entirely within a NGZ and thus is unboservable. In 84.09% of cases the change in information gain after 50 iterations was 0. More information was gathered in 15.55% and less information was gathered in 0.36% of cases. When the number of agents is low then it is may be impossible for the agents to be assigned to all of the POI, causing them to be spread out between many POI. Conversely when the number of agents is high, there may be too many agents for the number of POI. Each agent may observe one POI with other agents providing no information gain. If the region is saturated with agents before the Information Force System is run there may be little opportunity to improve the information gain.

5 Discussion and Related Work

The process of allocating agents to tasks is an active field of study. A formal analysis by Gerkey and Matarić created classifications for the different problem types of Multi-Robot Task Allocation (MRTA) (Gerkey and Matarić 2004). These classifications work for problems where the capabilities of the robots and the requirements of the tasks are well defined. A robot is either capable or incapable of performing a task. A task can be completed by one robot or requires multiple robots. The problem and solution we describe does not fit cleanly into these classifications. The UAVs may be considered single-task or multi-task robots depending on the sensors. The task of observing a point of interest may be a single-robot or multi-robot task depending on the information that could be gained. The analysis of algorithms by Gerkey and Matarić rely on assigning one or more agents to one or more tasks (Gerkey and Matarić 2004). Our problem and solution lack explicit assignments of agents to tasks which complicates this classification.

Other work has been done on assigning multiple robots to many tasks (Ravichandar, Shaw, and Chernova 2020; Robinson, Su, and Zhang 2021). Forming coalitions or teams based on the teams and the requirements of the tasks (Czatnecki and Dutta 2019; Aziz et al. 2021). These algorithms solve the problem of MRTA, but these require explicit assignment of agents to tasks. The positioning of an agent near a task is not considered useful to the utility functions in these algorithms. Additionally, the enforcement of geometric constraints, limiting the location of agents is not well defined for these algorithms. Constraints can be added to prevent certain agent task pairs, but these do not easily define if an agent at a particular position is capable of performing a task without modification to the algorithms.

Many of these techniques have been applied to UAVs with a variety of results (Ponda et al. 2015) and force based systems have been used on swarms for spacing drones (Gazi and Passino 2004). The combination of these algorithms to improve the efficiency and runtime has not been well explored. Our problem formulization and approach works well for distributing UAVs over a region to observe POI while avoiding NGZs. Other techniques can perform similar tasks such as creating paths to cover an area (Ponda et al. 2015), but these do not have the same flexibility as our approach.

6 Conclusion

Our approach effectively models and plans for geometric constraints and less strict agent tasking to maximize the information gain from the positioning of UAVs. Other approaches are limited by their design and implementation. Our Grid Placement algorithm and Information Force System allow for the geometric constraints which can model the region in which an agent can validly occupy. Our Agent-Location Paring algorithm builds on the Gale-Shapely algorithm to assign agents to locations that would maximize their information gain. These positions are further refined by the Information Force System. Through our experiments we demonstrated that these algorithms can be implemented and tested on large sets of locations and agents and have suggested strategies for increasing the computation speed. Continued work with this approach, the combination of multiple algorithms to increase the performance of the total system could provide promising results for a variety of tasking realted work. A limitation of the Agent-Location Paring algorithm is that it does not consider agents that have already been placed in the utility calculation. This can create situations when too many agents are assigned to a region of high information potential. A possible drawback is that recalculating the utility value of all locations after each pair is assigned could be costly. The partitioning improvement could be beneficial in this context. Further research is needed in the optimization of the information maximization.

7 Acknowledgments

This material is based upon work supported by the U.S. Army Combat Capabilities Development Command Aviation & Missile Center under Contract No. W911W6-19-C-0064. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the U.S. Army Combat Capabilities Development Command Aviation & Missile Center. Distribution Statement A, approved for public release. Distribution is unlimited (PR20221446).

References

Aziz, H.; Chan, H.; Cseh, Á.; Li, B.; Ramezani, F.; and Wang, C. 2021. Multi-robot task allocation–complexity and approximation. *arXiv preprint arXiv:2103.12370*.

Capezzuto, L.; Tarapore, D.; and Ramchurn, S. 2020. Anytime and efficient coalition formation with spatial and temporal constraints. In *Multi-Agent Systems and Agreement Technologies*. Springer. 589–606.

Czatnecki, E., and Dutta, A. 2019. Hedonic coalition formation for task allocation with heterogeneous robots. In *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*, 1024–1029. IEEE.

Gale, D., and Shapley, L. S. 1962. College admissions and the stability of marriage. *The American Mathematical Monthly* 69(1):9–15.

Gazi, V., and Passino, K. M. 2004. A class of attractions/repulsion functions for stable swarm aggregations. *International Journal of Control* 77(18):1567–1579.

Georgara, A.; Rodríguez-Aguilar, J. A.; and Sierra, C. 2021. Towards a competence-based approach to allocate teams to tasks.

Gerkey, B. P., and Matarić, M. J. 2004. A formal analysis and taxonomy of task allocation in multi-robot systems. *The International journal of robotics research* 23(9):939–954.

Ponda, S. S.; Johnson, L. B.; Geramifard, A.; and How, J. P. 2015. Cooperative mission planning for multi-uav teams.

Ravichandar, H.; Shaw, K.; and Chernova, S. 2020. Strata: unified framework for task assignments in large teams of heterogeneous agents. *Autonomous Agents and Multi-Agent Systems* 34(2):1–25.

Robinson, T.; Su, G.; and Zhang, M. 2021. Multiagent task allocation and planning with multi-objective requirements. In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*, 1628–1630.