Agent-based Sensor-Mission Assignment for Tasks Sharing Assets*

Thao Le
Department of Computing Science
University of Aberdeen
AB24 3UE, UK
thao.le@abdn.ac.uk

Timothy J. Norman
Department of Computing Science
University of Aberdeen
AB24 3UE, UK
t.j.norman@abdn.ac.uk

Wamberto Vasconcelos
Department of Computing Science
University of Aberdeen
AB24 3UE, UK
wvasconcelos@acm.org

ABSTRACT

A sensor network may be required to support multiple mission to be accomplished simultaneously. Furthermore, the environment may change at any time; i.e. a new mission may arrive at any time. In solving this many-mission, many-sensor problem in dynamic environments, conflicts between missions may occur for the use of sensor resources. A mechanism to match sensor resources to mission demands thus becomes necessary. In this paper, motivated by the conservation of resources, we consider the problem of sensor-mission assignment, in which sensors may be shared and reassigned between tasks. To achieve this, sensors are represented by agents, which coordinate to establish virtual organizations to meet mission requirements. The agent coordinating the achievement of a mission utilises a novel multi-round, Knapsack based algorithm, GAP-E, to allocate sensor agents to tasks based on bids received. Through simulations, we empirically demonstrate that this model provides a significant improvement in the number of completed missions as well as execution time.

Categories and Subject Descriptors

I.2.9 [**Computing Methodologies**]: Artificial Intelligence—*Sensors*

General Terms

Design, Measurement, Experimentation

Keywords

Sensors, Dynamic allocation, Resources sharing

1. INTRODUCTION

When a sensor network is deployed it is typically required to support multiple simultaneous missions. A given sensor may be beneficial to some missions, providing varying amounts of information to each one. Missions, on the other hand, can appear at any time and may place varying demands on sensors. In such multiple sensors and multiple missions problems in dynamic environments, conflicts between missions may occur for the use of the same sensor resources. Thus, we need efficient mechanisms to assign individual sensors to appropriate missions on the basis of information need.

An additional pragmatic problem arises in this domain. Due to the energy limitation and also to prolong the lifetime of the sensor network, conservation of energy consumed is an important consideration in managing wireless micro-sensor networks. However, to the best of our knowledge, in existing sensor-mission assignment approaches, each asset may be assigned to only one mission at any one time. The fact that assets may not be shared can waste energy since there might be more than one mission that requires the same kind of information which can be provided by a single sensor. Making decisions on how best to utilize limited sensor resources in order to satisfy mission demands without conflict and without wasting resources due to redundant assignment is, therefore, the key issue in sensor-mission assignment

Motivated by such necessity, we find that allowing sensors to be shared and to be reassigned between multiple tasks may provide substantial savings in sensor battery, often leading to improvements to the network's ability to meet its global objectives. In order to realize the idea, we need to decompose each mission to a set of specific tasks so that we can identify which tasks can share assets and which assets should or could be reassigned as circumstances change.

Within the model presented here, sensors are represented in the system by agents. These "sensor-agents" decide autonomously whether to offer to become involved in a mission depending on both their available resources and, in a broader sense, the environment that they are situated within. They communicate with each other, passing and sharing information when needed. In our context, they operate in a cooperative manner in which their resources are contributed toward achieving the global objective: the successful allocation of the most appropriate sensors to a mission according to its information requirements.

For each task, the sensor-agent located closest to its centre will act as the coordinator for that task; each task represents an information need within a geographical area at a certain

*The first author is a PhD student

time. The coordinator agent initiates interaction with other sensor agents within the scope of the task using a variant of the well-known contract net protocol [13]. The coordinator issues a call for bids for the delivery of information pertaining to the needs of the task. Each sensor agent receiving this call will analyze the information requirements of the task and make a bid only if it decides that it can satisfy the request based on its type and current workload. On the basis of the bids received, the coordinator agent utilises a novel multi-round, Knapsack-based algorithm, GAP-E, to allocate sensor agents to that task.

In this paper we make the following contributions to the state of the art. First, our solution allows the sensors to be shared between tasks, significantly improving the percentage of successfully allocated missions. Second, the search space of the problem and, consequently, the time consumed to find a solution, are greatly reduced. This increase the reliability of the system in practical situations.

The remainder of this paper is structured as follows: Section 2 details our approach including the novel GAP-E algorithm and how it is employed to provide our solution to the sensor-mission assignment problem in dynamic environments. In Section 3 we present a rigorous evaluation of our approach with varying sensor asset availability and varying mission arrival rates against existing solutions and an estimate of the optimal solution. Section 4 introduces the assignment problem within the broader context of the identification of information needs, sensor asset management, deployment and information delivery. We relate our model to existing research in this area, discuss the shortcomings of our model and point towards avenues for future research in Section 5, and, finally, we present our conclusions in Section 6.

2. PROPOSED APPROACH

Our proposed approach provides a solution for allocating a collection of intelligence, surveillance and reconnaissance (ISR) assets to a number of missions in order to satisfy the information requirements of that mission. These ISR assets are composed of various sensors, each with its own location and sensing range, and each sensor is able to provide different utilities to different tasks. We equate these sensors with agents, as each sensor is wrapped by an autonomous computational entity, that is, a software agent, which communicates with others by means of message-passing. A mission consists of a number of tasks, each task having a specific type, which, in turn, requires a number of sensor types and each task has a specified own location and operational range and has its own sensing demand. A task can only be satisfied if its demand is met (within a threshold) and all of its sensor types are present in its allocation. If a task is not satisfied, the mission requiring that task is not successful.

More formally, a sensor s_i is defined as the tuple $\langle \gamma_i, l_i, r_i, u_i \rangle$ where $\gamma_i \in \Gamma$ specifies s_i's type, l_i and r_i are the location and sensing range of s_i, u_i is the maximum utility s_i can provide in a single time unit, Γ is the set of all sensor types. A mission M is defined as a tuple $\langle T, l_m, r_m, time_m \rangle$ where T is a set of required tasks, l_m and r_m are M's location and operational range, $time_m$ is the time when M is active. Each task $t_j \in T$ is defined as the tuple $\langle \delta_j, l_j, r_j, d_j \rangle$ where $\delta_j \in \Delta, \delta_j = \{\gamma_k | \gamma_k \in \Gamma' \subseteq \Gamma\}$ denotes t_j's type, Δ is the set

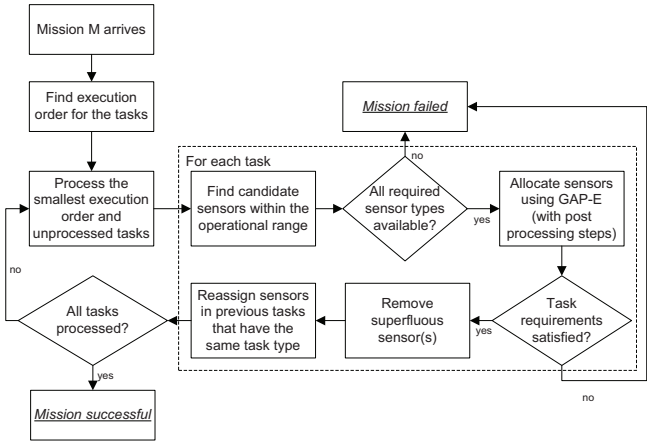

Figure 1: *Our proposed approach as a flowchart.*

of all task types, l_j and r_j specifies t_j's operational range (l_j is located inside (l_m, r_m)) and d_j is the sensing demand that t_j requires. The active time for t_j is the same as $time_m$ We denote u_{ij} as the utility that s_i can provide to t_j, which is defined as a percentage of u_i calculated by the ratio between the overlap of the ranges of s_i and t_j and the range of s_i. If the operational areas of s_i and t_j do not intersect, the value of u_{ij} will be 0.

Here we assume that the sensors cooperate with each other and they know both their locations as well as the location of other sensors. Moreover, we also allow a sensor to provide its service to multiple same-type tasks (for example, an audio sensor can provide the same information to all the detecting tasks that require its service). The sensor agents communicate with each other based on the message exchange protocol detailed in [7]. A mission can arrive at any time and there may be more than one mission active at any given time.

When a mission M arrives with its tasks, our approach will attempt to allocate all the sensors to the tasks as follows (see Figure 1):

1. The execution order of the tasks is established. Two tasks t_i and t_j belong to the same execution set (i.e. they can be executed at the same time) if their operational ranges do not intersect or their sensor type requirements do not overlap. If, however, two tasks have the same type, both will be in the same execution set. Initially, the execution set containing t_0 will be processed first followed by the set containing the next unprocessed task until all the tasks have been handled.

   It can be observed that the outcome of this algorithm depends on the order in which the tasks are executed. Obviously, it is impossible to determine in advance the execution order of the tasks in order to obtain the optimal outcome. However, to compliment the lack of available sensors for the tasks that are executed in subsequent orders, we allow the sensors to be shared and reassigned between these tasks. By having this feature, the tasks which are executed later can grab previously assigned sensors which, otherwise, will be unavailable.

2. For each task t_j:

   (a) Identify the available sensors within t_j's operation range and add to its sensor list. This is done by querying each sensor agent with a message containing the task information (task type, location) and waiting for an answer from that sensor[1]. Each sensor will analyze the task and decide to bid for the task, providing the amount of utility it can provide. Moreover, if a sensor has already been allocated to one or more tasks of the type δ_j, it can also provide a service to t_j. This will enable more tasks to be successfully allocated without reducing the practicability of the approach or putting more constraints on the sensors. After all the sensors within the operation range have been queried, if the available sensors do not cover the sensor requirements of t_j (i.e. a required sensor type cannot be found), then t_j cannot be allocated and the mission M fails.

   (b) If all required sensor types are available for the mission, the agent coordinating the tasks uses our multi-round GAP-E algorithm to find a potential allocation of sensors for t_j. The details of the GAP-E algorithm are presented in Section 2.1

   (c) After all the rounds are completed and all task requirements are satisfied, there is a final post processing step to release all the superfluous sensors (the one that can be released without violating t_j's requirements - both in terms of utility and sensor type). If there is more than one sensor which can be released, select the one with the smallest utility. This is to ensure that t_j will never be allocated more sensor agents than needed.

   (d) When the GAP-E finishes, if the final allocation does not satisfy t_j's requirements, M is failed. However, if t_j is successfully processed, the next step will attempt to reassign these allocated sensor to other tasks (not necessary from M) that are active.

3. When tasks in T have been processed, mission M is completed successfully.

2.1 The GAP-E Algorithm

In this section, we detail our algorithm, GAP-E, to allocate sensors to a particular task t_j. In order to realize the idea of stricter governing of selected sensors round-by-round, we introduce two additional matrices. P_j is the priority matrix that indicates the importance relationship amongst sensor types with regarding to t_j. C_j is the cost matrix that specifies the cost that will need to be met if t_j requires the service of a certain sensor. In addition, a budget b_j acts as a constraint that governs the number of sensors that can be allocated to t_j.

Figure 2 summarizes the steps that GAP-E takes to find an allocation to t_j, the details of which as follows:

[1]We do not consider failure cases in which, for example, sensor agents do not respond to such requests. There are, however, well known mechanisms for handling such situations such as "setting a deadline for receipt of responses" or simply add "with a deadline".

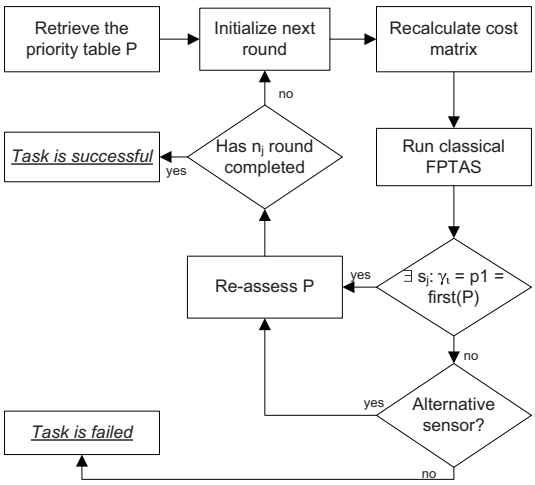

Figure 2: *GAP-E algorithm as a flow chart.*

1. Initially, we assume that t_j requires n_j sensor types ($|\delta_j| = n_j$), the priority matrix $P_j = \{p_i | i = 1...n_j\}$ where p_i is the sensor type that has the i^{th} importance with respect to t_j. This information is provided by the mission M. For example if t_j has the set of required sensor types $\delta_j = \{1, 2, 3\}$ and $P_j = \{3, 2, 1\}$ meaning that 3 is the most important sensor type and 1 is the least important one. Nonetheless, all these types must be presented in the final allocation otherwise the task will fail.

2. The utility matrix of the candidate sensors to t_j is updated. Based on the value of P_j, GAP-E introduces the concept of the cost matrix $C_j = \{c_{ij}\}$ for all s_i that has $u_{ij} > 0$. (c_{ij} is the cost of t_j using s_i's service as given in the bid received from s_i) and b_j as the overall budget of task t_j. Here, C_j has a similar objective as P_j; it is used to specify the relation between the sensor types and a particular task. The budget, b_j, is used to control the number of sensors allocated to t_j.

3. Next, there will be n_j rounds, with each round $r = 1..n_j$ composed of the following steps:

   (a) Reconstruct the matrix C_j so that if $k = p_i, l = p_j$ with $p_i, p_j \in P, i < j$ then $c_{k'j} < c_{l'j}\ \forall s_{k'}, s_{l'}$: $\gamma_{k'} = k, \gamma_{l'} = l$. If sensor type k is more important than type l then all the sensors of type k will have a lower cost than those of type l.

   (b) Run the FPTAS (Fully Polynomial Time Approximation Scheme) algorithm [14] with the input of $C_j, U = \{u_{ij}\}$ and budget $b = b_j * r/n_j - cost(A')$ with A' being the allocation from the last round. The obtained solution is then merged with A' to form the temporary allocation A.

   (c) If A does not contain at least one sensor of type p_1, we need to replace a sensor in A with a sensor of type p_1. Providing there exists some available sensors of type p_1 for t_j, the one with a minimum cost is selected as the target sensor. If, however, there are some sensors of type p_1 within t_j's range but they are all allocated to other tasks, GAP-E

will try to reassign sensors for one of such task in order to obtain a sensor of type p_1. In more details, if GAP-E can identify a task t_k that is currently holding sensor s_k of type p_1 and t_k can find a replacement sensor s_l without violating t_k's allocation requirements, GAP-E will reassign s_l to t_k and s_k is selected as the target sensor. If no target sensor can be found then t_j fails and consequently, M fails.

If adding this target sensor to A causes the allocation to go over-budget then we need to remove other sensors from A to accommodate this new sensor. This is done by repeatedly removing a sensor, which is (1) a sensor of type k that has more than two members within A or (2) a sensor of type l that has never been the first member of P in this or previous rounds. If more than one removable sensors can be identified, the one which provides the lowest utility is removed first. If the budget constraint is still violated but no removable sensor can be identified then t_j fails and consequently, M fails.

(d) Reassess P_j so that p_1 is now the sensor type that has the highest importance and does not appear in A. The order of $p_i|i = 2..n_j$ is kept as the initial version of P_j. If all the required sensor types are presented in A, P_j will revert back to the initial constructed version.

4. After n_j rounds has been completed, the GAP-E algorithm terminates.

In the following section, we present an example to illustrate the operations of this algorithm.

2.2 GAP-E Example

The following example demonstrates the workings of our approach. Let us assume that $t_1 = \{\delta_1, l_1, r_1, d_1\}$ where $\delta_1 = \{1, 2, 3, 4\}$ (sensor types 1, 2, 3 and 4 are required for task t_1) and $d_1 = 1.5$ (the task has a demand of 1.5). Additionally, there are the following candidate sensors within t_1's operational range: $S' = \{s_{11}, s_{12}, s_{21}, s_{22}, s_{31}, s_{41}, s_{42}\}$ with s_{ij} denoting sensor type i and index j. Here we have $U = [u_{111}, u_{121}, u_{211}, u_{221}, u_{311}, u_{411}, u_{421}] = [0.5, 0.25, 0.4, 0.7, 0.5, 0.15, 0.75]$ where u_{ij1} is the utility that s_{ij} can provide for t_1.

Initially, the budget for the task is set to $b_1 = 2$, and the initial priority matrix is $P_1 = [1, 2, 3, 4]$ (i.e sensor type 1 is most important, followed by type 2, etc.). The initial cost matrix is generated from the bids received from the candidate sensors as $C = [[c_{11}, c_{12}], [c_{21}, c_{22}], [c_{31}], [c_{41}, c_{42}]] = [[0.1, 0.15], [0.03, 0.17], [0.2], [0.05, 0.16]]$. There will be 4 rounds as follows:

- Round 1:
  - The matrix C_1 is recalculated based on C as $C_1 = [[c_{11}, c_{12}], [c_{21}, c_{22}], [c_{31}], [c_{41}, c_{42}]] = [[0.1, 0.15], [0.28, 0.42], [0.7], [0.8, 0.91]]$[2]
  - $b = 0.5$ (the total budget is initially split equally between the four sensor types required for this task), FPTAS returns $A = \{s_{11}, s_{12}\}$
  - Now we have all sensors of type 1, the priorities among tasks are now revised such that $P_1 = [2, 1, 3, 4]$. The allocation is within budget and so we do not consider sensors to be removed from the allocation. $cost(A) = 0.25$, $U(A) = 0.75$
- Round 2:
  - The matrix C_1 is recalculated as $C_1 = [[c_{21}, c_{22}], [c_{31}], [c_{41}, c_{42}]] = [[0.03, 0.17], [0.7], [0.8, 0.91]]$
  - $b = 1.0 - 0.25 = 0.75$, FPTAS returns $A = \{s_{21}, s_{22}\}$, combined with A' (the allocation from the previous round), we have $A = \{s_{11}, s_{12}, s_{21}, s_{22}\}$
  - Now we have sensors of type 1 and 2, thus $P_1 = [3, 1, 2, 4]$. The removal of sensors from the allocation need not be considered. $cost(A) = 0.45$, $U(A) = 1.85$
- Round 3:
  - The matrix C_1 is recalculated as $C_1 = [[c_{31}], [c_{41}, c_{42}]] = [[0.2], [0.8, 0.91]]$
  - $b = 1.5 - 0.45 = 1.05$, FPTAS returns $A = \{s_{42}\}$, combined with A' we have $A = \{s_{11}, s_{12}, s_{21}, s_{22}, s_{42}\}$
  - Now we have sensors of type 1, 2 and 4 but since $p_1 = 3$ and there is no type 3 in A, we will need to add a sensor of type 3 to A. Since there is only one sensor of such type, s_{31} needs to be added. However, adding s_{31} will cause the budget to be exceeded and we, therefore, will need to remove sensors from the allocation. Any of the sensors in A can be removed and thus, s_{12} is removed first since it has the lowest utility (u_{121}). By doing so, the cost of the solution falls below the budget and we have a valid allocation $A = \{s_{11}, s_{21}, s_{22}, s_{31}, s_{42}\}$ with $cost(A) = 1.41$ and $U(A) = 2.85$. Additionally, we have all the required sensor types, $P_1 = [1, 2, 3, 4]$
- Round 4:
  - The matrix C_1 is recalculated as $C_1 = [[c_{12}], [c_{41}]] = [[0.15], [0.8]]$
  - $b = 2.0 - 1.41 = 0.59$, FPTAS returns $A = \{s_{12}\}$, combined with A' we have $A = \{s_{11}, s_{12}, s_{21}, s_{22}, s_{31}, s_{42}\}$
  - As we have all the required sensor types, $P_1 = [1, 2, 3, 4]$ $cost(A) = 1.56$, $U(A) = 3.1$

Thus, GAP-E returns a valid allocation of $A = \{s_{11}, s_{12}, s_{21}, s_{22}, s_{31}, s_{42}\}$. However, since $U(A) = 3.1$ is greater than $d_1 = 1.5$, we will need to release superfluous sensors. Of the sensors in A, we can release any sensor in the set $\{s_{11}|s_{12}, s_{21}|s_{22}\}$. Thus, s_{12} will be released first because

[2]Since we have 4 sensor types and $P_1 = [1, 2, 3, 4]$, c_{11} and c_{12} remains the same, c_{21} and c_{22} are increased by $\frac{1}{4}$, c_{31} is increased by $\frac{1}{2}$, c_{41} and c_{42} are increased by $\frac{3}{4}$.

u_{211} is smallest of all (0.25). This brings the $U(A)$ down to 1.6. After removing s_{12}, $U(A) = 2.85$ and still greater than d_1. This time we can only release either s_{21} or s_{22} otherwise the sensor type requirements of t_1 will be violated. Thus, s_{21} will be released since $u_{211} < u_{221}$. As a result, $A = \{s_{11}, s_{22}, s_{31}, s_{42}\}$ is the final allocation to t_1.

Having defined our model, next section will details the results of our experiments.

3. EVALUATION

This section evaluates our approach in a range of different environments and assesses its performance in terms of the number of successfully completed missions and the amount of processing time required. There are a number of internal variables which control the behavior of our model as well as external variables which define the environment in which our model is being used. The system developed in Java allows us to manipulate these variables, conduct experiments and analyze the results.

3.1 Simulation Setup

Our approach is evaluated using randomly generated problems. A set of data is generated for each run. We compare the performance of our model (Multiple-Sensor-Mode or MSM) with the following alternatives:

1. Exclusive sensor mode (ESM): here each sensor can only be assigned to one task at a time. For each task, we test all combinations of sensors within the available candidates with the restriction that each sensor type only has one member[3]. The combination providing the highest utility value will be checked against the demand of the task, and, if acceptable, it will be selected as the final allocation for that task.

2. Shared sensor mode (SSM): this control operates in a similar way to ESM. The difference being that it allows sensors to be shared between two same-type tasks, which is exactly the feature that our approach also possesses. Again, the combination providing the highest utility will be selected as the final allocation if it satisfies the demand of the task in question. This is our implementation for the work presented in [9] (see section 5.

3. Shared sensor mode but without demand checking (SSM-NC): this control operates in a similar fashion to SSM. However, there is no evaluation against the task's demand. Instead, the combination providing the highest utility will be selected as the final allocation for that task. This will give us an idea of what the optimal success rate of the whole mission might be[4]

[3]Since the assignment problem is NP-Hard it is time- and memory-consuming to go through all possible combinations. Thus one single sensor per type is chosen to reduce the running time of this case.

[4]This cannot be considered to be the optimal result since it does not involve checking all the combinations of all the potential allocations for each individual task and then measuring their requirements. There are cases where optimal allocation involves non-local maxima allocation. Given the set of experiments need to be carried out, however, it is impractical to do a complete exhaustive search to find the optimal solution.

We recall from Section 2 that a mission is composed of a set of individual tasks and can it only be satisfied if all of its tasks can be allocated. Here, the mission arrival rates are controlled by the *rate_per_hour* parameter, which ranges from 2 to 8, and *number_of_days* parameter, which is kept at 2 days. Each mission can last for an arbitrary amount of time, ranging from 5 minutes to 4 hours.

There are *total_sensor_types* different sensor types, which will vary between 4 and 8 and, for each sensor type, there will be *total_sensors_per_types* sensors. For each mission, the number of tasks will be varied between 3 and 8. There will be *total_task_types* different task types, which will vary between 4 and 8. Each task type will require a number of different sensor types, which is varied between tasks and has the figure randomized between 1 and 4. These individual sensor type requirements are generated randomly and have the value between 1 and *total_sensor_types*.

The battlefield has the size of 400m x 400m. This is where the sensors and missions are deployed in uniformly random locations. Each sensor range (r_i) is randomized between 20m and 40m and their maximum utility is calculated as $(r_i/40)^2$, which ensure the values between 0.25 and 1. The operation range of the tasks are set to be randomized between 40 and 80m. We now turn to the specific results.

3.2 Results

HYPOTHESIS 1. *MSM performs well in comparison to the estimated optimum.*

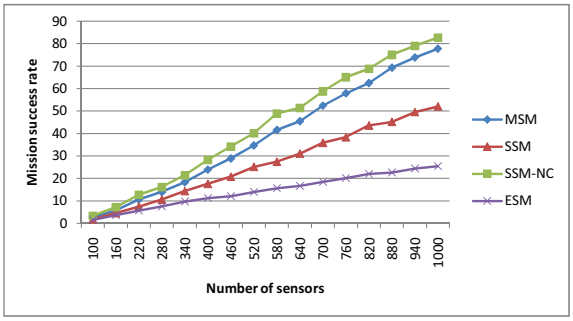

Figure 3: *Mission success rate with 4 sensor types and 4 missions arriving per hour.*

EVALUATION. Figure 3 shows the mission success rate of the four mechanisms with *total_sensor_types* = 4, *rate_per_hour* = 4 and *total_ sensors_per_types* between 25 and 250. Figure 4 shows the mission success rate with *total_sensor_ types* = 8, *rate_per_hour* = 8 and *total_sensors _per_types* also between 25 and 250. As the overall objective is to increase the number of successful missions, this control variable strongly indicate the performance of each mechanism.

As can be seen from both Figures 3 and 4, SSM provides better results than ESM. This is because in SSM, multiple same-type tasks can share a single sensor and, thus, a task in SSM has a greater chance of being successfully allocated.

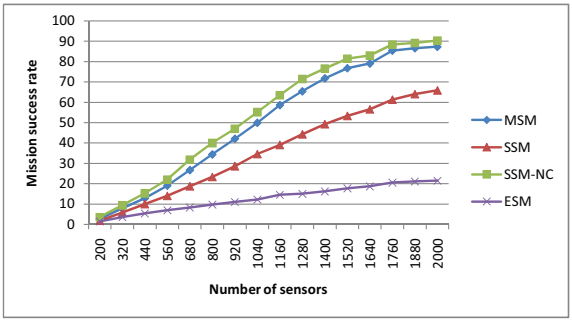

Figure 4: *Mission success rate with 8 sensor types and 8 missions arriving per hour.*

However, in both SSM and ESM, only one sensor per sensor type can be presented in the allocation and in many cases this is not sufficient to satisfy the sensing demand of a task. This explains why MSM has a significantly better mission success rate than SSM and ESM. With SSM-NC, since the demand requirement is not checked, the number of successful missions is the largest. However, this does not reflect the optimum allocation since there are situations in which the sensor type requirements of a task can be met but the sensing demand cannot be achieved. Nonetheless, the assignment problem being identified in this paper is NP-hard and it is impractical to do an exhaustive search of all combinations to find the optimal solution. Thus, the SSM-NC control is introduced here to give an idea of where the optimal solution might lie.

HYPOTHESIS 2. *The computational complexity (running time) of MSM is much less than that of other mechanisms.*

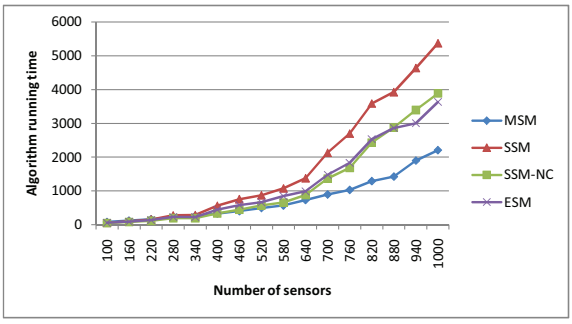

Figure 5: *Running time (ms) with 4 sensor types and 4 missions arriving per hour.*

EVALUATION. Similar to hypothesis 1, figure 5 shows the running time of the four mechanisms with *total_sensor_types* = 4, *rate_per_hour* = 4 and *total_ sensors_per_types* between 25 and 250. Figure 6 shows the running time with *total_sensor_ types* = 8, *rate_per_hour* = 8 and *total_sensors _per_types* also between 25 and 250. Of course, the running time of each mechanism depends largely on the machine that the experiments are run on. However, putting them all together can give a picture of their overall complexity.

As can be seen from both figures, MSM running time is always the smallest. Both SSM and SSM-NC has to exhaustively search through all the potential combinations of sensors to find the best allocation, their complexity will increase exponentially when the number of sensors increased. Thus, their running time are always highest and in many cases can be 2.5 times larger than that of our MSM. On the other hand, MSM employs the FPTAS algorithm (see section 2.1) which has the polynomial complexity and therefore, its running time only increases steadily when the number of sensors increases.

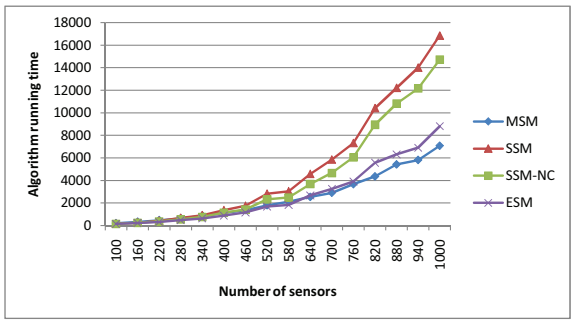

Figure 6: *Running time (ms) with 8 sensor types and 8 missions arriving per hour.*

HYPOTHESIS 3. *The computational complexity of MSM is increased in a steadily fashion with the number of missions (or tasks).*

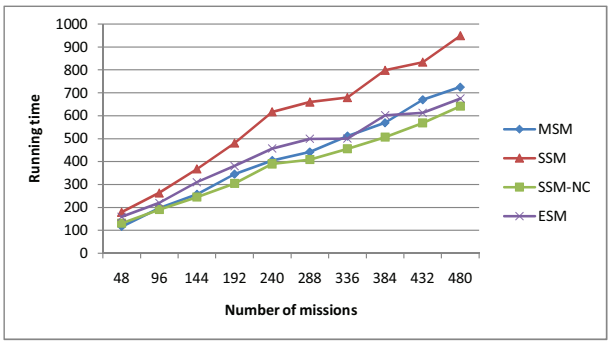

Figure 7: *Running time (ms) with 4 sensor types and 25 sensors per type.*

EVALUATION. To evaluate this hypothesis, we measure the running time of the mechanisms with *total_sensor_types* = 4, *total_sensors _per_types* = 25 and varies *rate_per_hour* between 1 and 10. The result is displayed in figure 7. It is clearly that as the number of mission increases from 48 to 480, the running time of MSM also increases steadily from 100 to just over 700ms. As our GAP-E algorithm has a polynomial complexity depending on the number of sensors and missions, once we keep the former variable unchanged, therefore, MSM running time increases inline with the number of missions.

4. SENSOR ASSIGNMENT IN CONTEXT

This section discusses the allocation problem addressed in this paper in a specialized environment setting. In particular, the problem of sensor-mission assignment is defined as

that of allocating a collection of intelligence, surveillance and reconnaissance (ISR) assets (both sensors and platforms) to one or more missions. Missions are composed of various tasks focused on satisfying their information requirements (IRs). These IRs will be identified as part of the process of mission planning. IRs are derived from questions such as "is there suspicious activity on the main supply road?" (see Figure 8). Each IR is then broken down to a set of scenario-specific informal requirements (SSIRs) such as "are there suspicious vehicles on the road?" or "is there suspicious pedestrian activity along the road side?". Before being able to match these to sensing types, decision-makers identify the interpretation tasks (ITs) which indicate what kinds of things need to detected, identified, distinguished, etc. The results of this further breakdown resemble a set of database queries like "detect vehicles where vehicle type or behaviours is suspicious", "detect people where person type or behaviours is suspicious". Furthermore, information is typically available that details the ISR assets (platforms and sensors), characterised in terms of their types, locations, readiness status, etc., that can be deployed to meet the information requirements.

Once having identified the ITs, given the informal of ISR assets available in the theatre, semantic matchmaking mechanism (such as those used by SAM [5]) may be employed to identify appropriate types of assets for the interpretation tasks specified. These approaches provide decision-makers with an at-a-glance view of feasible solutions.

It is this kind of information regarding information needs that we assume is available for the mission-sensor assignment problem. We are, therefore, concerned with the problem of moving from a "type level" fitting to an "instance level" allocation. When a mission needs to be accomplished, we consider a set of real assets available in the field that are compatible with the corresponding sets of sensor types required for each mission that arrives.

Allocated assets will then be configured for deployment in the operating environment. As the sensor network operates, information will be disseminated and delivered to users and the operational status of the assets will be monitored, thus closing the loop between the specification of information requirements and sensor information delivery. Both the ongoing monitors and the appearance of new tasks and ISR requirements can cause the decision-makers to reassess the sensor-mission assignment solution.

5. DISCUSSION & RELATED WORK

Sensor-task assignment problems in wireless sensor networks have been studied mainly using simplified models in which only a single sensor type is introduced and the exclusive of resources is required. For example, in [4, 8] the authors propose distributed approaches to solve the assignment problem assuming that the same type of sensors are deployed in the battlefield and that there is no competition for the sensing resources between tasks. On the other hand, the decentralised approaches considered in [12, 1, 7] introduce the competition between sensing tasks of same type. However, this approach set constraints which prevent more than one sensor from being assigned to any one task.

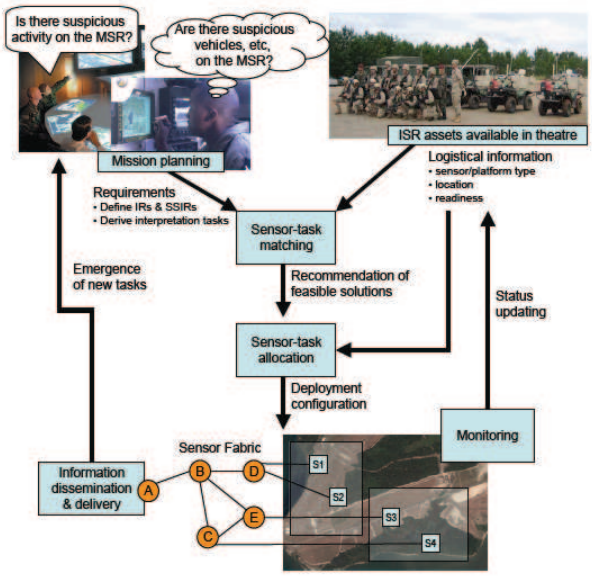

Figure 8: *The sensor-mission assignment problem in context.*

Our proposed E-GAP algorithm presented in Section 2 is an adaptation of the MRGAP algorithm proposed in [7]. The MRGAP algorithm aims to solve the static assignment problem, which is a common generalisation of the problems presented in [3, 1, 11], incorporating both budgets and a profit thresholds. The idea of that algorithm is to consider missions as knapsacks that together form an instance of the Generalized Assignment Problem (GAP). The author of [3] give an approximation algorithm for GAP which takes a knapsack algorithm as a parameter. There, the standard knapsack algorithm FPTAS [14] offers an approximation guarantee of $2 + \epsilon$. MRGAP, however, still lacks the ability to consider multiple sensor types. Moreover, it does not take into account the trade-off between communication cost and utilities gained. Nonetheless, the battery life-time of individual sensor is typically limited by the power required to transmit their data. As a result, power conservation issues in wireless sensor networks are essential and attract the interest of many researchers. In [10], for example, the authors selected a sensor acting as a mediator to relay data for other sensors. It saves the battery life of other sensors to the detriment of its own battery. To choose such mediator, the authors proposed a payment scheme in which the power p to reliably transmit over a distance d (p is proportional to the square of d) is considered as the decision value.

A number of protocols to locally make decisions for a sensor in a wireless sensor network have begun to tackle the challenge of coordinating between the network's interconnected nodes given the absence of a central coordinator. Protocols in [10, 6] allow sensors to request other sensors to forward data, which provides sensors with a broader range of information, often leading to an improvement in the network's ability to meet its global objectives. However, it requires extra communication that imposes an energy cost on the network. Therefore, the authors of [2] observed that the proper level of coordination (the degree of hops a sensor

broadcast message to) leads to a significant increase in the performance of the network.

Protocols have also been developed in [12, 9] where each task leader runs a local protocol to match sensors within two hops to the requirement of the task. Additionally, these works are the most important related to ours. They consider the many-sensor type, many-task type assignment problem. They also use the results of matching sensor type to mission to reduce the search space in order to find the allocation. However, they need to generate and check all the instances of feasible solutions of the matching sensor type problem. This is not good in dense sensor network. The main difference with our work is that the authors do not allow assets to be shared between tasks.

There are several shortcomings of our model that may restrict its applicability. First, we do not take into account the fact that some of the sensors can be mounted on mobile platforms. This is typical in many situations; for example, an image sensor can be mounted on an unmanned aircraft. Thus, such sensor will have a greater operational area compared to a static one. We plan to address this problem by incorporating new utility prediction model that will include the cost of relocating sensors within the battlefield.

Second, the tasks comprising a mission are independent of each other. In practice, there are situations in which there exist inter-dependencies between themselves. For example, there might be tasks that can only be allocated based on the results of some other tasks, which mean they can only be processed afterward. Furthermore, in such cases, the time-line for these tasks will be different from each other (rather than derived from the mission time-line). Future work will need to include a suitable planning model in order to be able to handle such scenarios.

Finally, only static sensors are considered in this model. Our next target research will consider mobile sensors that can be mounted on a platform. In order to achieve this goal, we need to incorporate a joint utility model instead of the current additive one and consequently, complete the resource sharing feature. Specifically, we aim to be able to handle situations in which a task only needs a percentage of the capacity of a sensor, the rest will be still available for other tasks.

6. CONCLUSION

In this paper we have introduced an agent-based (and hence decentralised) approach to solving the sensor-mission assignment problem for tasks sharing assets. We transformed the global sensor-mission assignment problem into a collection of sub-problems of sensor-task assignment. These sub-problems are then solved by task coordinator agents employing our GAP-E algorithm, which utilizes the multi-round knapsack algorithm to provide concrete solution consisting of the required sensor types together with the specific sensors belonging to these types. Our approach has been evaluated in a number of different scenarios, and we have demonstrated empirically that good results can be achieved in a considerably less time compared to the traditional exhaustive search counterpart.

7. REFERENCES

[1] A. Bar-Noy, T. Brown, M. P. Johnson, T. F. L. Porta, O. Liu, and H. Rowaihy. Assigning sensors to missions with demands. In M. Kutylowski, J. Cichon, and P. Kubiak, editors, *ALGOSENSORS*, volume 4837 of *Lecture Notes in Computer Science*, pages 114–125. Springer, 2007.

[2] E. Bulut, J. Zheng, Z. Wang, and B. K. Szymansk. Analysis of cost-quality tradeoff in cooperative ad hoc sensor networks. In *ACITA 2008*, 2008.

[3] R. Cohen, L. Katzir, and D. Raz. An efficient approximation for the generalized assignment problem. *Information Processing Letters*, 100(4):162–166, 2006.

[4] C. Frank. Algorithms for generic role assignment in wireless sensor networks. In *in SenSys Š05: Proceedings of the 3rd International Conference on Embedded Networked Sensor Systems*, pages 230–242, 2005.

[5] M. Gomez, A. Preece, M. P. Johnson, G. Mel, W. Vasconcelos, C. Gibson, A. Bar-Noy, K. Borowiecki, T. Porta, D. Pizzocaro, H. Rowaihy, G. Pearson, and T. Pham. An ontology-centric approach to sensor-mission assignment. In *EKAW '08: Proceedings of the 16th international conference on Knowledge Engineering*, pages 347–363, 2008.

[6] D. B. Johnson. Scalable and robust internetwork routing for mobile hosts. In *In Proceedings of the 14th International Conference on Distributed Computing Systems*, pages 2–11, 1994.

[7] M. P. Johnson, H. Rowaihy, D. Pizzocaro, A. Bar-Noy, S. Chalmers, T. F. L. Porta, and A. Preece. Frugal sensor assignment. In S. E. Nikoletseas, B. S. Chlebus, D. B. Johnson, and B. Krishnamachari, editors, *DCOSS*, volume 5067 of *Lecture Notes in Computer Science*, pages 219–236. Springer, 2008.

[8] H. Park and M. B. Srivastava. Energy-efficient task assignment framework for wireless sensor networks. Technical report, 2003.

[9] A. Preece, D. Pizzocaro, K. Borowiecki, G. de Mel, M. Gomez, M. Vasconcelos, A. Bar-Noy, M. P. Johnson, T. L. La Porta, H. Rowaihy, G. Pearson, and T. Pham. Reasoning and resource allocation for sensor-mission assignment in a coalition context. In *MILCOM 2008*, 2008.

[10] A. Rogers, E. David, and N. R. Jennings. Self-organized routing for wireless microsensor networks. *IEEE Transactions on Systems, Man and Cybernetics, Part A*, 35(3):349–359, 2005.

[11] H. Rowaihy, M. Johnson, T. Brown, A. Bar-Noy, and T. L. Porta. Assigning sensors to competing missions. In *Technical Report NASTR-0080-2007*, Network and Security Research Center, Department of Computer Science and Engineering, Pennsylvania State University, University Park, PA, USA, 2007.

[12] H. Rowaihy, M. Johnson, D. Pizzocaro, A. Bar-Noy, T. L. Porta, and A. Preece. Sensor-task assignment protocols. In *International Technology Alliance, Tech. Rep. (submitted for publication)*, 2008.

[13] R. G. Smith and R. Davis. Frameworks for cooperation in distributed problem solving. pages 61–70, 1988.

[14] V. V. Vazirani. *Approximation Algorithms*. Springer, 2001.