RC25115 (W1011-164) November 30, 2010
Computer Science

IBM Research Report

A Flexible Solutions Methodology for Resource Allocation with Applications to Sensor Network Operations

Srikanth Hariharan
ECE Department
The Ohio State University
Columbus, OH 43210

Chatschik Bisdikian
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598

Lance M. Kaplan, Tien Pham
U.S. Army Research Laboratory
Adelphi, MD 20783

Research Division
Almaden - Austin - Beijing - Cambridge - Haifa - India - T. J. Watson - Tokyo - Zurich

A Flexible Solutions Methodology for Resource Allocation with Applications to Sensor Network Operations*

Srikanth Hariharan† Chatschik Bisdikian‡ Lance M. Kaplan, Tien Pham§

ABSTRACT

Motivated by the need to judiciously allocate scarce sensing resources to attain the highest benefit for the applications that sensor networks serve, in this paper, we develop a flexible solutions methodology for maximizing the overall reward attained subject to constraints on the resource demands under fairly general reward or demand functions. We map a broad class of related problems into an integer programming problem and provide an iterative Lagrangian relaxation technique to solve it. Each iteration step involves solving for a maximum weight independent set of an appropriately constructed graph, which, in many cases, can be obtained in polynomial time. We apply our methodology to the problem of tracking targets moving over a period of time through a non-homogeneous, energy-constrained sensor field. With rewards represented by the quality of information attained in tracking, we study its trade-offs and relationship with energy consumption and periodic measurement taking. We further illustrate how to apply our methodology to an entirely different problem of how an unmanned air vehicle must traverse through a network of unattended ground sensors such that it maximizes the information collected from these sensors under delay constrains.

1. INTRODUCTION

In a wireless sensor network, sensors with multiple modalities can be used to estimate a variety of features from objects of interest. For example, in target tracking, a radar can be employed to estimate the location and velocity of a target, while an imaging sensor can be employed to estimate its physical dimensions. Furthermore, fusing the information collected from the different modalities provide us with a more complete and accurate description providing a richer quality of information (QoI) [3], such as reducing the uncertainties regarding the tracks of the targets sensed.

Sensor networks can potentially perform multiple sensing and processing tasks at will. However, due to their limited bandwidth, energy, and computing resources, it becomes imperative to design and operate them in a way that is respectful of these limitations by judiciously allocating their resources to the tasks at hand. It is, thus, a goal of our work to develop a flexible solutions methodology (the "framework") for maximizing the reward obtained by allocating resources to tasks subject to constraints on the resource demands.

Resource allocation problems are typical applications of integer programming [4] and our framework falls in this category. However, with our interest to the problem rooted in sensor networks, our framework specifically focus on ensuring that the reward that is obtained by, say, fusing information from multiple sources can be represented as a general function of the rewards obtained from individual sources. This is of particular importance in heterogeneous, and multi-modal sensor networks. For example, the reward obtained by fusing information from an imagery sensor (e.g., camera) and an acoustic sensor may not be a sum (or even a weighted sum) of the rewards obtained by the individual sensors. Further, our framework takes into account demand constraints from non-homogeneous sensors, where, for example, a camera may require more energy than an acoustic sensor.

We show that without demand constraints, the problems at hand map to maximum weight independent set problems. Though, in general, these problems are NP-Hard, we also show that we can find such sets in polynomial time in a number of cases because of the structures of the graphs in which they are obtained. When

*Research was sponsored by the US Army Research Laboratory and was accomplished under Agreement Number W911NF-06-3-0002-P00013. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the US Army Research Laboratory, or the U.S. Government. The US Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

†Srikanth is with the ECE Dept., The Ohio State University, Columbus, OH 43210, USA, *harihars@ece.osu.edu*.

‡Chatschik is with the IBM T. J. Watson Research Center, Hawthorne, NY 10532, USA, *bisdik@us.ibm.com*.

§Lance and Tien are with the U.S. Army Research Laboratory, Adelphi, MD 20783, USA, {*lance.m.kaplan,tien.pham1*}*@us.army.mil*.

demand constraints are present, a particularly challenging problem in integer programming, we use Lagrange multipliers to formulate the dual problem and provide an iterative solution, which again involves maximum weight independent sets.

Our framework constitutes a novel and flexible use of a Lagrangian-based integer programming solution methodology to sensor networks, where it finds a number of applications. We study two such (entirely different) applications, a multi-target tracking case, and a case where an unmanned air vehicle (UAV) traverses a sensor network collecting information from unattended ground sensors (UGSs). Our framework can be directly applied to the multi-target tracking problem, and we extensively study the convergence properties of our algorithms, and the trade-offs between the QoI and a number of system parameters such as the energy utilized by the system, the period of sensing, and the number of sensors. For the UAV case, we show how a minor modification of our framework results in a problem that maximizes the information collected from the UGS subject to deadline constraints. This modified framework allows the UAV to visit a sensor multiple times before the deadline, and also a reward function that allows a general relationship between the amount of information possessed by a sensor, and the delay for the UAV to collect information from it.

Considering the operation of a system over slotted time, our main contributions in this paper are:

- the development of a general integer programming framework for allocating a constrained pool of resources to one or more tasks in each slot, with a system-level reward function comprising a general function of the rewards from individual resources for each task;
- the use of the framework to model a class of problems as a *maximum weight independent set* problem having a polynomial complexity for a large number of cases;
- an iterative solution for the integer programming problem using a primal-dual gradient projection algorithm;
- the application of the framework to the efficient operation of sensor networks including multi-target tracking with non-homogeneous sensors, and delay-constrained, flight-path scheduling of a UAV collecting data from UGSs; and
- extensive performance, simulation, and trade-off evaluation while tuning system parameters for improving the QoI for the multi-target tracking problem.

The rest of the paper is organized as follows. In Section 2, we describe the system model. In Section 3, we put forth our framework for assigning resources to multiple tasks. In Section 4, we develop an iterative algorithm to solve the problems formulated in Section 3, and discuss analytical results on optimality and convergence. In Section 5, we quantify our analytical results through extensive numerical evaluations for the multi-target tracking problem. In Section 6, we provide various additional applications of our framework, and detail the case of the UAV collecting information from UGSs in a delay-constrained manner. In Section 7, we discuss related work and finish in Section 8 with some concluding remarks.

2. SYSTEM MODEL

We consider a system comprising a pool of *resources* S of size N and a pool of *tasks* T of size M, where a group of resources from S is allocated to perform (or service) a group of tasks from T. Each such allocation has a demand and a reward associated with it, where for example, the demand could represent the energy consumed by a group of sensors (the resources) for tracking a group of targets (the tasks). The reward could represent the QoI, e.g., the variance of the tracking estimators obtained by such an allocation. Various objectives and constraints may be considered such as maximizing the total reward subject to a constraint on the total demand, or minimizing the total demand subject to a constraint on the total reward. There could also be additional constraints such as a particular group of tasks must obtain at least a certain amount of reward.

For the case of sensor networks, we further consider time-slotted operation with resource allocations occurring at each slot. In this case, we refer to the time slots during which a sensor takes measurements as *sampling instants* or *sampling periods*. An example of our system model for a target tracking application is provided in Figure 1. The black sensors track multiple targets during the same sampling instant, the blue sensors track only one target, and the red sensor tracks no targets during that sampling instant.

3. PROBLEM MODEL AND FRAMEWORK

In this section, we start with a general problem formulation and in the next section we present its solutions methodology. We consider two problem models, a *task-oriented model* (TOM), and a *resource-oriented model* (ROM) and discuss the applicability conditions for each. We then describe a number of extensions that these models can handle. In this (and the next) section, we focus on allocations happening in a single time slot only, and, hence, no time index will be used. In later sections, we consider applications of these models on sensing systems running over a number of slots.

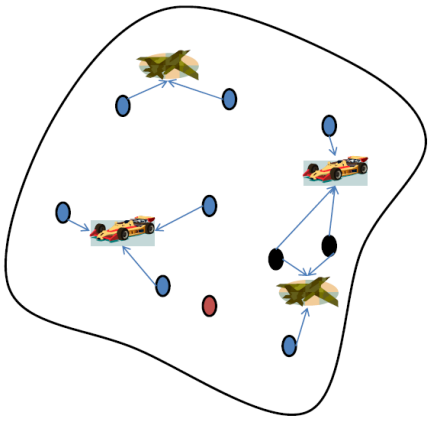


Figure 1: Model

3.1 Task-Oriented Model (TOM)

According to this model, a task is assigned to a group of resources. Specifically, for each task $i \in T$, $i = 1, \ldots, M$, we associate a collection of sets $\mathbb{K}_i$ that represents the possible groups of resources that can be assigned simultaneously to that task. Let $\mathbb{K}_i = \{K_1^i, K_2^i, \ldots, K_{m_i}^i\}$ where for each $l \in \{1, \ldots, m_i\}$, $K_l^i \subseteq \{0\} \cup S$. The "0" element represents the case that task i is not assigned to any resource (during a slot). Clearly, the "0" element can be a member of only one of the sets in $\mathbb{K}_i$ and that set must be a singleton. The introduction of the $\{0\}$ set in $\mathbb{K}_i$ allows us to explicitly model the penalty for not assigning task i to any resource.

For each $i \in T$, and $j \in \mathbb{K}_i$, we define x_{ij} to be the *assignment indicator* variable:

$$x_{ij} = \begin{cases} 1, & \text{if task } i \text{ is assigned to } \textit{all } (\textit{and only}) \\ & \text{the resources in the set } j; \text{ and} \\ 0, & \text{otherwise.} \end{cases} \quad (1)$$

Let q_{ij} represent the reward obtained from task i when $x_{ij} = 1$ holds true, i.e., when task i is assigned to (and serviced by) *all* the resources in the set $j \in \mathbb{K}_i$ (and only in this set). Likewise, let e_{ij} denote the resource demand for task i when $x_{ij} = 1$ holds true, and let e be the total demand constraint on the system. Now the optimization problem Π_T for TOM can be formulated as follows:

Problem Π_T:

$$\text{maximize} \sum_{i \in T} \sum_{j \in \mathbb{K}_i} q_{ij} x_{ij} \quad \text{s.t. (1)} \sum_{j \in \mathbb{K}_i} x_{ij} = 1 \text{ for each } i \in T;$$

$$\text{(2)} \sum_{i \in T} \sum_{j \in \mathbb{K}_i} e_{ij} x_{ij} \leq e; \text{ and}$$

$$\text{(3)} \; x_{ij} \in \{0, 1\}. \quad (2)$$

The objective of Π_T is to maximize the sum of the rewards obtained by all the tasks in the system. Constraint (1) is a matching constraint and it states that each task i must be assigned to exactly one of the sets of resources in $\mathbb{K}_i$, which by construction contains all permissible alternatives for assigning task i to the resources (even the $\{0\}$ set). Note that the reward q_{ij} obtained when resources in the set $j \in \mathbb{K}_i$ perform task i can be an *arbitrary* function of the rewards obtained from each resource separately. Constraint (2) is the demand constraint. This constraint implies that if a single resource is assigned to multiple tasks, the total demand required by this resource must be the *sum* of the individual demands for performing each task. Therefore, the demand function cannot be arbitrarily chosen here.

When considering sensor networks, because of the aforementioned arbitrary relationship between q_{ij} and the per-resource rewards, TOM can be used even when, say, the QoI (the reward) attained by fusing information from multiple sensors (the resources) is not restricted to be the sum of the individual QoIs as was the case in [9], [15]. For this reason, we may also refer to this model as the *Fusion-Oriented Model*. This objective function also allows modeling correlations among sensor readings related to the same task, such as a specific target multi-target tracking.

3.2 Resource-Oriented Model (ROM)

According to this model, a resource is assigned to a group of tasks. In analogy to TOM, for each resource $i \in S$, $i = 1, \ldots, N$, we associate a collection of sets $\mathbb{J}_i$ that represents the possible groups of tasks that can be assigned simultaneously to the resource. Let $\mathbb{J}_i = \{J_1^i, J_2^i, \ldots, J_{n_i}^i\}$ where for each $l \in \{1, \ldots, n_i\}$, $J_l^i \subseteq \{0\} \cup T$. The "0" element represents the case that resource i is not assigned to any task. Clearly, the "0" element can be a member of only one of the sets in $\mathbb{J}_i$ and that set must be a singleton. The introduction of the $\{0\}$ set in $\mathbb{J}_i$ allows us to explicitly model the penalty for not assigning resource i to any task.

Defining x_{ij}, q_{ij}, and e_{ij} correspondingly to their TOM counterparts, the optimization problem Π_S for ROM can be formulated as follows:

Problem Π_S:

$$\text{maximize} \sum_{i \in S} \sum_{j \in \mathbb{J}_i} q_{ij} x_{ij} \quad \text{s.t. (1)} \sum_{j \in \mathbb{J}_i} x_{ij} = 1 \text{ for each } i \in S;$$

$$\text{(2)} \sum_{i \in S} \sum_{j \in \mathbb{J}_i} e_{ij} x_{ij} \leq e; \text{ and}$$

$$\text{(3)} \; x_{ij} \in \{0, 1\}. \quad (3)$$

While TOM and ROM appear structurally identical, there are important semantic differences between the two. For instance, with ROM, the demand required by a resource when assigned to multiple tasks can be an *arbitrary* function of the demand required for individual tasks. This can be easily seen from the fact that we associate a separate demand e_{ij} for every different

set j of tasks to which resource i is assigned. This can cover cases, where, for example, an acoustic sensor may not require any additional energy to track more targets in its vicinity, but a radar might need to use more energy to track multiple targets. This aspect is captured by this constraint, and for this reason, we also refer to this model as the *Demand-Oriented Model*. On the other hand, ROM can only model the overall reward obtained from a task as the *sum* of the individual rewards from each resource that is assigned to it. Therefore, the use of this model is not recommended when, say, a fusion function for information from multiple sensors is arbitrary.

Table 1 summarizes the differences between these models. Note that due to their respective structures, TOM can accommodate correlations in the rewards obtained from each resource assigned to a given task, while ROM can accommodate correlations in the rewards obtained by each task performed by a given resource.

Attribute	TOM	ROM
Reward	Arbitrary	Sum
Demand	Sum	Arbitrary
Correlations	Among resources for a given task	Among tasks for a given resource

Table 1: Differences between the task (TOM) and resource (ROM) oriented models

Note that the analyses developed in [15], [6], [10] are all special cases of the frameworks developed in this section.

3.3 Extensions

We focus on TOM here; similar extensions can be made for ROM as well.

A resource can perform only one task but a task can be serviced by multiple resources

We can model this constrained problem by including the following linear constraint in problem Π_T:

$$\text{for each resource } k \in S, \quad \sum_{i \in T} \sum_{\{j \in \mathbb{K}_i : k \in j\}} x_{ij} \leq 1 \quad (4)$$

The meaning of this constraint is that for all sets j for which a given resource k is a part of, at most one of the resource assignment indicators x_{ij} can equal 1. However, it is possible that multiple tasks are not assigned to any resource. So $k = 0$ is not included in this constraint. We study this extension in our multi-target tracking application example in Section 5.

***Minimize the total demand subject to minimum reward* q**

This problem is the dual problem of Π_T. The solution to this problem can be found by combining a binary search to the solution of Π_T. The algorithm is given below.

1. Initialize $e_{left} = 0$ and $e_{right} = e_{max}$, where e_{max} is the maximum demand required by the system (when all the resources are used). Set $e = \frac{1}{2}(e_{left} + e_{right})$.
2. Solve Π_T with demand constraint e. If the reward obtained is greater than q, set $e_{right} = e$; else, set $e_{left} = e$. Set $e = \frac{1}{2}(e_{left} + e_{right})$.
3. Repeat Step 2 until $e_{right} - e_{left}$ is less than a threshold.

The reason this algorithm minimizes the total demand is because the objective function of Π_T cannot decrease as the amount of resources used by the system increases.

Guarantee that certain tasks achieve at least a given minimum reward

Suppose that task i requires a reward of at least q. The solution to this problem is the following.

1. For each $j \in \mathbb{K}_i$, if $q_{ij} < q$, set $x_{ij} = 0$.
2. Solve Π_T with this modification.

Since with TOM, the reward obtained by a task is explicitly given by the set $j \in \mathbb{K}_i$ assigned to it, by removing the sets that provide a reward that is lower than the constraint, we can guarantee a minimum reward for that task.

Guarantee that certain resources satisfy a given maximum demand

Suppose that resource k can satisfy a demand of at most e_k. This can be modeled by the following linear constraint:

$$\sum_{i \in T} \sum_{\{j \in \mathbb{K}_i : k \in j\}} x_{ij} e_{ij}^{(k)} \leq e_k, \quad (5)$$

where $e_{ij}^{(k)}$ is an arbitrary function of e_{ij} representing the demand contribution of resource k in the set j of sensors. Note that e_{ij} represents the demand required by *all* the resources in the set j to perform the task i: $e_{ij} = \sum_{l \in j} e_{ij}^{(l)}$. Individual resources could have required less demand. This problem can be solved in the same manner (using a Lagrange multiplier for this constraint) as Π_T is solved in the next section.

4. SOLUTIONS METHODOLOGY

In this section, we develop iterative algorithms for problems Π_S and Π_T, and discuss optimality and convergence results. We first find the optimal solution for the problem without demand constraints. Then, we use this to find an optimal solution for the problem with demand constraints.

4.1 Without Demand Constraints

We show that an optimal solution can be obtained by finding a *maximum weight independent set*. An *independent set* in a graph is a set of nodes no two of which have an edge between them. A maximum weight independent set is an independent set with maximum total weight of nodes. Finding a maximum weight independent set in a general graph is NP-Hard–it follows from the fact that finding a maximum clique in a general graph is NP-Hard [11]. Nonetheless, in many of our cases, due to the structure of the graphs involved, it is not NP-Hard to find such a set.

THEOREM 4.1. *Optimal solutions to the problems Π_S and Π_T (without demand constraints) can be obtained by finding a maximum weight independent set in the graph $\mathcal{G}$ constructed as follows:*

- *for each variable x_{ij}, create a node (i,j) in $\mathcal{G}$;*
- *for each node (i,j) in $\mathcal{G}$, associate a weight q_{ij}; and*
- *for every two nodes (i_1,j_1) and (i_2,j_2), create an edge between them if and only if $i_1 = i_2$.*

PROOF. From the above construction, two nodes (i_1,j_1) and (i_2,j_2) in $\mathcal{G}$ can both be in an independent set of $\mathcal{G}$ if and only if $i_1 \neq i_2$. This implies that the variables $x_{i_1j_1}$ and $x_{i_2j_2}$ can both equal 1 simultaneously if and only if $i_1 \neq i_2$. Therefore, for TOM, an independent set cannot consist of any two nodes in $\mathcal{G}$ that correspond to the same task. Similarly, for ROM, an independent set cannot consist of any two nodes in $\mathcal{G}$ that correspond to the same resource. Thus, the constraints are satisfied.

Since by finding a maximum weight independent set, we maximize $\sum_{i\in T}\sum_{j\in\mathbb{K}_i} q_{ij}x_{ij}$ for TOM, and $\sum_{i\in S}\sum_{j\in\mathbb{J}_i} q_{ij}x_{ij}$ for ROM, while satisfying the constraints, this maximum weight independent set provides an optimal solution to both problems Π_S and Π_T (without demand constraints). □

COROLLARY 4.1. *When a resource can perform only one task (in a given time slot), an optimal solution to TOM (without demand constraints) can be obtained by finding a maximum weight independent set in the graph $\mathcal{H}$, where $\mathcal{H}$ is constructed as follows:*

- *construct $\mathcal{G}$ as in Theorem 4.1; and*
- *for any two nodes (i_1,j_1) and (i_2,j_2) where $\{0\} \notin j_1$ and $\{0\} \notin j_2$, create an edge between them if and only if $j_1 \cap j_2 \neq \emptyset$.*

PROOF. In the case where a resource can perform only one task, we have all the original constraints in problem Π_T (apart from the demand constraint), and the additional constraint given in (4). $\mathcal{G}$ represents all the constraints except the constraint in (4). Since for any node (i_m,j_m), j_m represents the set of resources performing task i_m, if $j_1 \cap j_2 \neq \emptyset$, one of the resources in j_1 and j_2 is performing both tasks i_1 and i_2. Therefore, we create an edge between these two nodes. Further, we do not create an edge in $\mathcal{G}$ if j_1 and j_2 do not have any common resource. Therefore, an independent set in $\mathcal{H}$ will satisfy the required constraints, and hence a maximum weight independent set in $\mathcal{H}$ will provide an optimal solution for this case in the absence of demand constraints. □

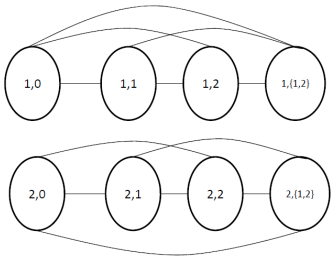


Figure 2: The graph $\mathcal{G}$ is a disjoint union of cliques

We now show that a maximum weight independent set in $\mathcal{G}$ can be found in polynomial time.

THEOREM 4.2. *$\mathcal{G}$ is a union of disjoint cliques, and hence a maximum weight independent set in $\mathcal{G}$ can be found in polynomial time.*

PROOF. By the construction of $\mathcal{G}$ in Theorem 4.1, two nodes (i_1,j_1) and (i_2,j_2) will have an edge between them if and only if $i_1 = i_2$. Hence, there is no edge between these nodes when $i_1 \neq i_2$. Therefore, the set of nodes having the same first label i_1 form a clique, and, thus, $\mathcal{G}$ is a union of cliques. Since there are no edges between any node in a clique and any node in another clique, $\mathcal{G}$ is a union of disjoint cliques.

Hence, a maximum weight independent set in $\mathcal{G}$ is obtained by simply selecting the node in each clique with maximum weight. This can clearly be obtained in polynomial time. □

Figure 2 illustrates the result in Theorem 4.2 for the case of two resources and two tasks. The figure is applicable to both TOM and ROM; for example, for TOM the node label $(1,\{1,2\})$ means that task 1 is assigned to both resources 1 and 2, while for ROM, the same node label means that resource 1 performs both tasks 1 and 2. It can be easily seen that $\mathcal{G}$ is a disjoint union of cliques.

In Figure 3, we provide an example of the special case where a resource can perform only one task. The graph $\mathcal{H}$ is not a disjoint union of cliques. For instance, we can see that there is an edge between $(1,\{1,2\})$ and $(2,1)$ since if task 1 is assigned to both resources 1 and 2 (in TOM), then task 2 cannot be assigned to resource 1. It

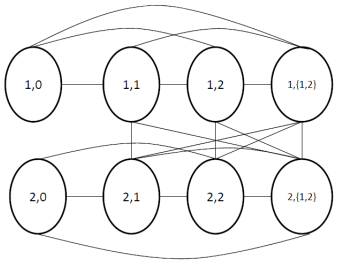


Figure 3: Graph $\mathcal{H}$: A resource can perform only one task

is NP-Hard in general to find a maximum weight independent set in this case. However, when the number of tasks is a constant, the complexity of finding a maximum weight independent set in this graph is polynomial in the number of nodes in the graph. The reason is as follows: Since we can only select one group of resources for each task, if there are k tasks in the system, and n nodes in this graph, the complexity of finding a maximum weight independent set in this graph is $O((n/k)^k)$.

4.2 With Demand Constraints

We now take the demand constraints into account as well. We only present the solution for TOM; the solution for ROM is identical. Associating a Lagrange multiplier λ for the demand constraint, the dual objective function for Π_T can be obtained as follows:

$$D(\lambda) = \max_{(2.1), x_{ij}\in\{0,1\}} \Big\{\sum_{i\in T}\sum_{j\in\mathbb{K}_i}(q_{ij}-\lambda e_{ij})x_{ij}\Big\} + \lambda e, \quad (6)$$

where the (2.1) qualifier for the "max" operator refers to constraint (1) in the maximization formulation for Π_T in (2). From (6), it can be immediately seen that for a given λ, $D(\lambda)$ can be obtained by finding a maximum weight independent set in $\mathcal{G}$ where the weight of each node (i,j) in $\mathcal{G}$ (see Theorem 4.1) is modified from q_{ij} to $(q_{ij} - \lambda e_{ij})$.

The dual optimization problem of Π_T is given by

$$\min_{\lambda\geq 0} D(\lambda). \quad (7)$$

We can solve this dual optimization problem using the following gradient projection algorithm.

Algorithm 1:

1. Initialize $\lambda = 0$.
2. At iteration k, compute $\vec{x}^{(k)} = \{x_{ij}^{(k)} : i \in T, j \in \mathbb{K}_i\}$ as

   $$\vec{x}^{(k)} = \arg\max_{(2.1), x_{ij}\in\{0,1\}} \Big\{\sum_{i\in T}\sum_{j\in\mathbb{K}_i}(q_{ij}-\lambda^{(k-1)}e_{ij})x_{ij}\Big\}.$$

   This can be computed by finding a maximum weight independent set in $\mathcal{G}$ with the weights modified as described before.
3. At iteration k, update λ as follows.

   $$\lambda^{(k)} = \big[\lambda^{(k-1)} + \alpha^{(k)}(\sum_{i\in T}\sum_{j\in\mathbb{K}_i} e_{ij}x_{ij}^{(k)} - e)\big]^+, \quad (8)$$

   where $[y]^+ = \max\{0, y\}$. The coefficient $\alpha^{(k)}$ is a positive step-size used at iteration k and it can be chosen according to Theorem 4.3 later on. One of the possible choices of $\alpha^{(k)}$ is $1/k$.
4. Stop when $\lambda^{(k)} - \lambda^{(k-1)} < \gamma$, where γ is a threshold.

It follows from step 3 that λ increases when the demand required by the system is greater than e, and it decreases when the demand required by the system is less than e. Hence, a binary search algorithm can be used instead for updating λ as well. This algorithm results in much faster convergence than the gradient projection algorithm. However, the gradient projection algorithm is very useful when multiple Lagrange multipliers need to be updated simultaneously (corresponding to multiple constraints).

Note that we can also easily modify the algorithm of finding a maximum weight independent set to find, instead, a minimum weight independent set by subtracting each of these weights from a large number (that is greater than all the weights in the graph). We can then find an assignment such that a task i is assigned to at least one of the groups of resources in $\mathbb{K}_i$ for TOM (correspondingly, a resource i is assigned to at least one of the groups of tasks in $\mathbb{J}_i$ for ROM). This gives the required solution to this problem. A minimum weight independent set is useful when the objective is to minimize, say, the sum of the errors over all the tasks. Note that this is also a representation of the reward obtained.

4.3 Computational Complexity

We consider the binary search algorithm for updating λ. The number of iterations required by the binary search algorithm is $O(\lceil\log_2(d/\gamma)\rceil)$, where γ is the threshold in Algorithm 1, and $d = e_{max}$ is the maximum demand that the system can utilize, i.e., the system uses all the resources available.

(i) *Resources perform multiple tasks*: The complexity of finding a maximum weight independent set in this case is simply the number of nodes in the graph $\mathcal{G}$. Assume that there are s resources, t tasks, and that at most k of the resources can be combined for a task, e.g., information from k out of s sensors can be fused for a sensing task. Then the number of nodes in $\mathcal{G}$ for TOM is at most $t \times \left(1 + s + \binom{s}{2} + \ldots + \binom{s}{k}\right) \sim O(ts^k)$. Further, since this computation needs to be performed at most $\lceil\log_2(d/\gamma)\rceil$ times (for binary search), the overall complexity is given by $O(ts^k\lceil\log_2(d/\gamma)\rceil)$.

(ii) *Resources perform one task*: In this case, the complexity of finding a maximum weight independent

set is $O(s^{kt})$; the k is as in the case (i) above. Therefore, the overall complexity with demand constraints is $O(s^{kt}\lceil \log_2(d/\gamma)\rceil)$.

4.4 Optimality and Convergence

If the x_{ij}'s were relaxed to take continuous values in $[0,1]$, then for diminishing step-sizes ($\alpha^{(k)} \to 0$, $\sum_{k=1}^{\infty} \alpha^{(k)} \to \infty$), as $k \to \infty$, Algorithm 1 would converge to an optimal solution for both the primal and dual problems, and there would be no duality gap [2]. However, since the x_{ij}s are indicator ($\{0\text{-}1\}$-valued) variables, we would need post-processing to obtain valid values for the x_{ij}'s. It is, in general, very hard to analytically compare this post-processed integer solution with the optimal integer programming solution.

With our framework, we do not relax the integer variables x_{ij}. In this case, while the dual program is a linear optimization program for λ, the primal program is an integer optimization program. Therefore, there could potentially exist a duality gap. The following result shows the convergence of λ.

THEOREM 4.3. *For any $\epsilon > 0$, for diminishing step-sizes ($\alpha^{(k)} \to 0$, $\sum_{k=1}^{\infty} \alpha^{(k)} \to \infty$), $\exists B > 0$ such that $\forall k > B$, $|\lambda^{(k)} - \lambda^*| < \epsilon$, where λ^* is the optimal solution to the dual problem.*

The proof follows from [8].

While λ converges to the optimal solution of the dual problem, we cannot guarantee that the primal objective function converges. Suppose that the primal objective function does not converge. When the algorithm stops, we will obtain two values of λ, λ_1 and λ_2, where, say, $\lambda_1 < \lambda_2$, and $\lambda_2 - \lambda_1 < \gamma$. λ_1 will result in an infeasible solution to the primal problem, while λ_2 will result in a feasible solution to the primal problem. This has been proven in [1, 12] and we also observed it during our numerical evaluations in Section 5.2. One way to find the primal optimal solution is to perform a branch and bound starting with these solutions that we obtained from Algorithm 1. Recently, it was shown in [12] that given a ρ-approximation algorithm for Step 2 of Algorithm 1, a $(\frac{\rho}{\rho+1} - \epsilon)$-approximation algorithm for the primal problem can be obtained using the feasible and infeasible solutions obtained either by binary search or by Algorithm 1. In our case, since some of the maximum weight independent sets can be solved in polynomial time because of the graph structure, using Algorithm 1 and the algorithm provided in [12], we can obtain a $(0.50 - \epsilon)$-approximation for our primal problem.

5. EXAMPLE APPLICATION: MULTI-TARGET TRACKING

In this section, we first provide an example of how to apply our framework to a typical sensor-based multi-target tracking scenario. We then study a multi-target tracking scenario focusing on the system trade-offs (along various system parameters such as energy, number of sensors, QoI, etc.) and the framework's convergence performance. In these examples, sensors and the energy they consume correspond to resources and resource demands, respectively; also, tracking targets and QoI attained correspond to tasks and rewards received, respectively.

5.1 A Multi-target Example Scenario

Suppose that there are $M = 2$ targets to be tracked. The sensor field comprises $N = 3$ sensors: a radar R and two acoustic sensors A_1 and A_2. In order to estimate the location of a target, we need information from at least one radar, or at least two acoustic sensors. The measurement from a radar can also be fused with the measurement from two acoustic sensors. So information can be fused either by two acoustic sensors, or by a radar and two acoustic sensors. If the targets are moving along a straight line, the radar can use a linear Kalman filter to estimate the location. The acoustic sensors can get the direction of arrival estimate from the targets. One could relate the QoI for the radar with the variance of the Kalman filter estimates and the QoI for the acoustic sensors with the log-likelihood ratio of the corresponding estimates.

Suppose that a sensor can sense only one target. The energy required to obtain fused information can depend not only on the energy required for sensing by the individual sensors but also on the energy required for communicating information between the sensors. Therefore, the energy required for fused information could be modeled by a super-additive expression. Also, the QoI obtained by fusion can be modeled by an arbitrary function of the QoIs obtained by individual sensors.

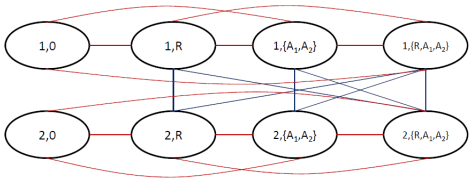


Figure 4: Multi-Target Tracking Example

For this system scenario, let us now construct the graph $\mathcal{H}$. The number of nodes in $\mathcal{H}$ is given by 2 (# of targets) $\times$ { 1 (target sensed by no sensor) + 1 (for the radar) + 1 (# of ways of selecting two acoustic sensors) + 1 (for a combination of the radar and two acoustic sensors) } = 8. The edges are shown in Figure 4. The edges representing the constraint that a target can be tracked by at most one group of sensors are shown in red, while those representing that a sensor cannot track more than one target are shown in blue. It is possible

to track both the targets if the radar is assigned to one target, and the two acoustic sensors are assigned to the other target. It is also possible to track just one target if all the three sensors are assigned to the same target. These solutions correspond to two different independent sets in $\mathcal{H}$, and we can find the optimal solution by finding the independent set with maximum total weight.

5.2 Numerical Results

We now investigate a multi-target tracking problem with Kalman filters and study the performance of the QoI obtained; we use the variance of the track estimate as the QoI. As we mentioned before, any filter can be used, any fusion technique can be applied, and any QoI metric can be used. We use Kalman filters here for the purpose of illustrating how our framework can be applied. Even with these basic filters, we obtain insightful results on how QoI behaves with energy. We consider the case where a sensor can track only one target during a sampling instant, but a target can be tracked by multiple sensors.

5.2.1 Setup

We a assume as system with $M = 3$ targets and $N = 9$ sensors, three of which are high energy sensors and six are low energy. Information is required from two low energy sensors, or one high energy sensor, or a combination of one high energy sensor and two low energy sensors in order to estimate the location of a target. The mobility model for the targets are given by the following equations, where $k = 0, 1, 2, \ldots$ is the slot index (we use a scalar formulation for ease of presentation):

$$x_i(k+1) = a_i x_i(k) + v_i(k), \quad i \in \{1, 2, 3\}, \tag{9}$$

where $v_i(k)$ is AWG process noise with distribution $\mathcal{N}(0, Q_i)$. The measurement model for each of the high energy and low energy sensors are is (we drop the sensor index for brevity):

$$z_i^e(k) = x_i(k) + y_i^e(k), \quad e \in \{h, l\}, \text{ and } i \in \{1, 2, 3\}, \tag{10}$$

where h and l stand for high and low energy, respectively, and $y_i^e(k)$ is AWG measurement error with distribution $\mathcal{N}(0, R_i^e)$. We assume that R_i^e, $e \in \{h, l\}$, depends on the distance between the sensor and the target. Specifically, the measurement error variance is 4 for a high energy (h) sensor and 9 for a low energy (l) sensor, if the distance between the sensor and the target is less than 10 units; the measurement error variance increases by 1 if the distance is between 10 and 20 units, and increases by 2 if the distance is more than 20 units.

The locations of the sensors and the initial locations of the targets are random. Table 2 provides the numerical values of the parameters in the setup. e_h, e_l, e_{ll}, and e_{hll} represent the energies used by a high energy sensor only, a low energy sensor only, a combination of two low energy sensors, and a combination of one high energy and two low energy sensors, respectively.

Parameter	Value	Parameter	Value
a_1	1.5	Q_1	4
a_2	1.2	Q_2	6
a_3	1.1	Q_3	4
R_h	4	e_h	2
R_l	9	e_l	1
e_{hll}	6	e_{ll}	2.4

Table 2: Simulation Parameters

Let $P_i^p(k)$ represent the one-step predicted variance for target i at slot k, and $P_i^c(k)$ represent the corrected variance after the measurement for target i at slot k is received. The variance update of the Kalman filters are given by the following equations, $k = 1, 2, \cdots$ ($i = 1, 2, 3$):

$$P_i^p(k) = a_i^2 P_i^c(k-1) + Q_i, \tag{11}$$

$$P_i^c(k) = P_i^p(k) - G_i(k) P_i^p(k), \tag{12}$$

where $G_i(k)$ is the Kalman filter gain and equals $P_i^p(k)/(P_i^p(k) + R_e)$, $e \in \{h, l\}$. Substituting the gain back to (12) yields:

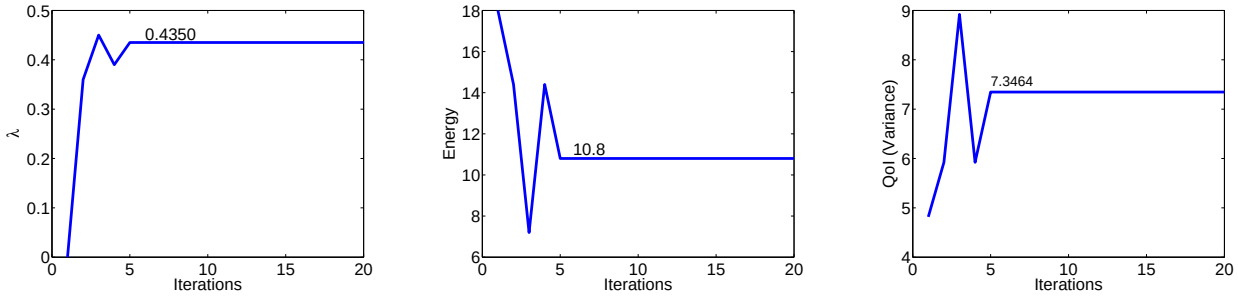
$$P_i^c(k) = \frac{P_i^p(k) \cdot R_i^e}{P_i^p(k) + R_i^e} \Leftrightarrow \frac{1}{P_i^c(k)} = \frac{1}{P_i^p(k)} + \frac{1}{R_i^e}. \tag{13}$$

Thus, when "knowledge" is added, which in this case is the newly arriving measurement with variance R_i^e, the tracking variances before and after incorporating the new knowledge exhibit a harmonic relationship; this is a classical result from estimation theory [16]. We adopt a similar relationship for the QoI obtained when fusing across track estimates from different sensors. One could use alternative fusion expressions but these would not alter the application of our framework, and hence we do not consider them in the context of illustrating our framework.

With this setup, and interested in minimizing the sum the QoI (variance) over all the targets, we now study a number of performance metrics such as the convergence of our algorithms, the performance of the overall variance with energy, and over a time horizon.

5.2.2 Algorithm Convergence

We start with an experimental study of the convergence for the algorithms developed in Section 4. Note that while the Lagrange multiplier always converges, the primal objective function may not converge. This, however, does not mean that the primal objective function will diverge to $\pm\infty$, but rather that it will oscillate within ϵ of the optimal solution for some $\epsilon > 0$. We study two cases, one in which the primal objective function converges, and one where it does not.



(a) λ (b) Total Energy (c) Variance

Figure 5: Instance where the primal objective function converges

We first consider a total energy constraint of 10.8. The values of the total energy used by the system at each iteration, the Lagrange multiplier λ, and the value of the primal objective function (variance) are shown in Figure 5. We initialize $\lambda = 0$. We use Algorithm 1 for this experiment, and we choose the step-size at iteration k to be $\alpha^{(k)} = 0.05/k$ and γ is set to 10^{-4}. When $\lambda = 0$, there are no constraints on the energy, and hence the network uses the maximum possible energy available (which equals 18, as can be easily derived). Since this is greater than 10.8, λ increases. As λ keeps increasing, the network starts using less energy. Ultimately, it converges to using a total energy equal to the constraint (=10.8). Since the QoI is the variance here, when the system uses higher total energy we get a lower variance. These can be seen in Figure 5. We see that the QoI, the energy, and λ all converge within 5 iterations.

We now consider a total energy constraint of 10. We run the same algorithm as before. We observe from Figure 6 that while λ still converges, the total energy used by the system, and the primal objective function do not converge. In fact, the total energy oscillates between 7.2 and 10.8. One reason that this happens here is because there is no optimal solution that exactly uses a total energy of 10 while there are solutions that use a total energy of 7.2 and 10.8 respectively. Note that 7.2 is a feasible solution while 10.8 is an infeasible solution. Therefore, we can use the two values of λ corresponding to these energies, and determine the optimal solution using other techniques discussed in Section 4. In this case, it turns out that the optimal solution uses a total energy of 7.2 (which corresponds to using all six low energy sensors but no high energy sensor). Further, we can see that our algorithm reaches a total energy of 7.2 within 5 iterations.

5.2.3 *QoI vs. Energy and Sampling Period*

Energy can be saved by a sensor network not only by reducing the total energy used for taking a measurement but also by varying the time period over which a measurement is taken. Here, we still consider a single sampling instant. However, the period of sampling is changed. When the period of sampling is i, the mobility equations for the targets can be modified as follows.

$$x_j(k+i) = a_j^i x_j(k) + a_j^{i-1} v_j(k) + ... + v_j(k+i-1), \quad (14)$$

where $j = 1, 2, 3$. The variance of a target's mobility is given by $Q_j(i) = Q_j * (a_j^{2i} - 1)/(a_j^2 - 1)$.

Figure 7 shows how the overall variance (QoI) changes with energy for different sampling periods. We notice that the overall variance decreases as energy increases. Note that when the total energy constraint is less than 6, the overall variance is much higher than otherwise. This is because one or more targets cannot be tracked when the total energy that can be used is less than 6. The drop in the overall variance is much lower when the total energy constraint varies from 6 to 18. When the total energy constraint is increased above 18, there is no change in the overall variance. This is because the maximum energy that can be used by the system is 18. Therefore, we do not get any additional information by further increasing the energy.

We also note that as the sampling period increases, the overall variance also increases. While there is a significant change in the variance when the sampling period is increased to 2 from 1, the difference in the overall variance between sampling periods i and $i+1$ decreases as i increases. These results can be used to appropriately select a sampling period for the required application.

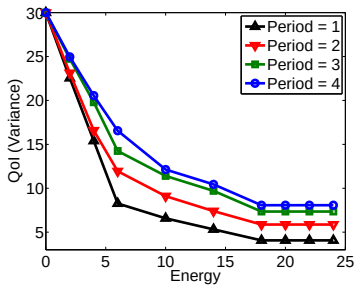
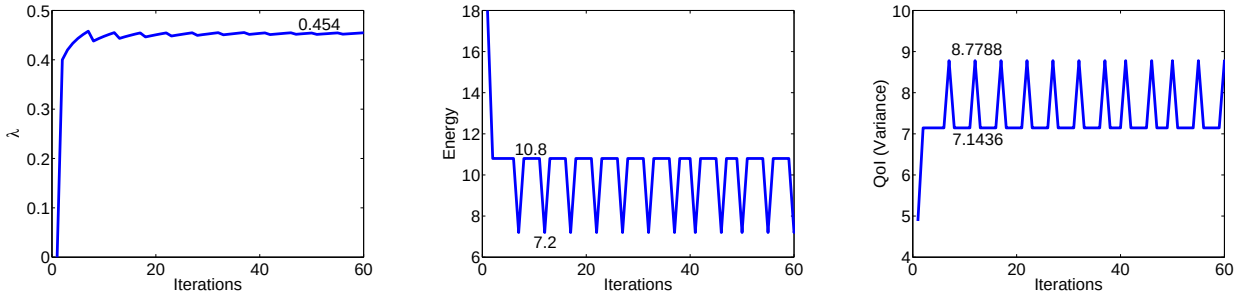


Figure 7: QoI vs. Energy and Sampling Period



(a) λ (b) Total Energy (c) Variance

Figure 6: Instance where the primal objective function does not converge

5.2.4 *QoI over a Time Horizon*

Finally, we study how the QoI behaves over a time horizon for different energies and different sampling periods. The targets move according to their mobility models described in the setup. When we do not make a measurement during a time slot (due to the sampling period chosen), we use the predicted variance of the Kalman filter as the QoI. When we make a measurement, we use the corrected variance of the Kalman filter to represent the QoI.

Figure 8 shows the performance of the system over a time horizon for four different values of energy, and three different sampling periods. Note that the energy here is the average energy used over the time horizon. When the sampling period is i and the average energy used is e_{avg}, then the energy used during the sampling slot is ie_{avg}, and zero during all other slots. From all the figures, we can observe that as the energy increases, the overall variance decreases and hence we get a better QoI. When the sampling period is one, we can see from Figure 8(a) that the overall variance converges in a few time slots. For sampling periods greater than one, we observe that the predicted variance is much higher than the corrected variance. Therefore, when we do not take a measurement we get a high overall variance. This is the reason that we see an oscillating function for higher sampling periods.

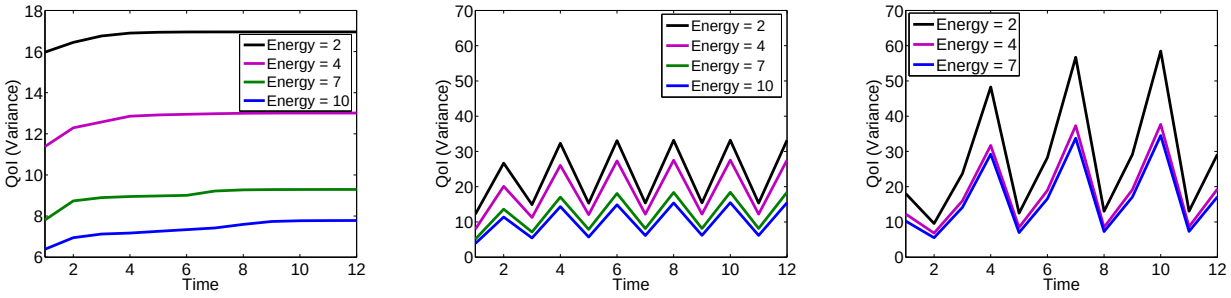
Figure 9(a) represents the QoI averaged over the time horizon for various values of the total energy. It shows that as the sampling period increases, the average QoI becomes worse. This is because of the fact that the predicted variance is much higher than the corrected variance. Therefore, if we have a choice of reducing energy consumption by either increasing the sampling period or decreasing the energy, we should opt to decrease the energy since a higher sampling frequency achieves a lower time-averaged variance for a given time-averaged energy.

Figure 9(b) shows the behavior of the QoI over the time horizon for various sampling periods when the average energy over the time horizon is fixed at 7. This figure is particularly interesting because we observe that for the time slots at which we take measurements, the QoI for a higher sampling period actually performs better than the QoI for a lower sampling period. This is because for a higher sampling period, we can use higher energy while still maintaining the average energy at 7. However, because the maximum total energy that can be used by the system is fixed (18 in our case), we cannot arbitrarily keep increasing the sampling period and still get improving QoI. When the sampling period is 3, we use an energy of 21 (which is greater than the maximum total energy available in the system). Therefore, when the sampling period increases above 3, the variance will again start increasing. For instance, we numerically observed that the variance obtained by the first measurement for sampling period 3 was 5.5151 while that for sampling period 4 was 5.5707. *For a given maximum energy e_{max}, we observe that there is a critical sampling period at which the QoI obtained during a sampling slot is minimum. This will depend on the system parameters in general. Here, it can be calculated as $\lceil e_{max}/e_{avg} \rceil$.*

Finally, we compute a ratio of the average QoI attained until the current time slot to the average energy used until the current time slot, and study how it behaves over the time horizon (Figure 9(c)) when the average energy used over the entire time horizon is 7. We observe that ultimately the high value of the predicted variance again dictates the behavior of this metric. For a higher sampling period, we get a higher ratio.

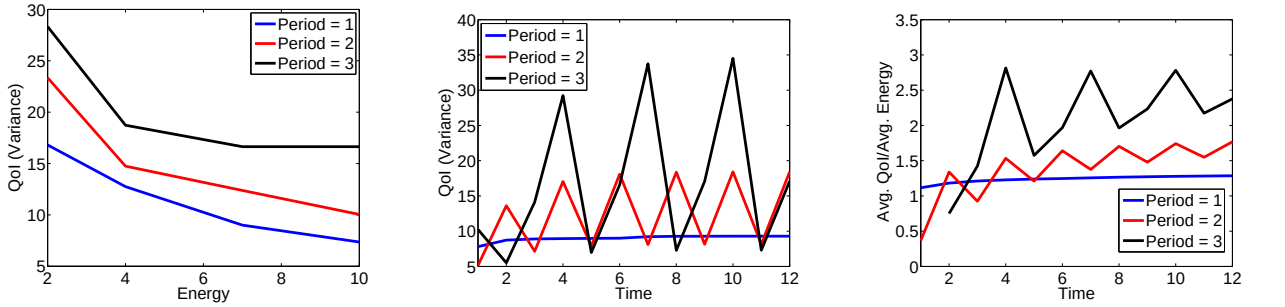
5.2.5 *QoI vs. No. of Sensors*

In this experiment, we modify the original setup, and study how the QoI varies with the number of sensors, the number of targets, and the level of fusion. We assume that a sensor can simultaneously track multiple targets. Each sensor uses a bank of Linear Kalman Filters to compute an estimate of the predicted variance for each target (which represents the QoI here). There are no energy constraints in this experiment. The measurement error variance for a sensor is chosen randomly



(a) Sampling Period = 1 (b) Sampling Period = 2 (c) Sampling Period = 3

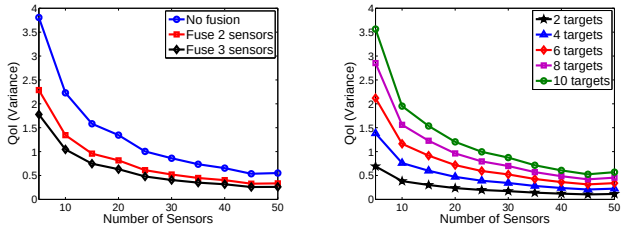
Figure 8: QoI over a time horizon



(a) Average QoI vs. Energy (b) QoI vs. Time (c) Avg. QoI/Avg. Energy vs. Time

Figure 9: Various QoI metrics

between 0 and 6. The experiment is repeated 300 times, and the results are averaged.



(a) No. of Targets = 5 (b) Level of Fusion = 3

Figure 10: QoI vs. No. of Sensors

From Figure 10, we see that as the number of sensors is increased, the total variance decreases. This is because as the number of sensors increases, there are more, on the average, sensors with lower measurement error variances to track targets. However, note that the drop in variance is significantly lower when the number of sensors is increased from 15 to 50 (say) than from 5 to 15. Therefore, in many applications, it may not be necessary to have a high density of sensors to track a group of targets.

In Figure 10(a), we fix the number of targets to 5. We vary the levels of fusion, i.e., the maximum number of sensors whose information can be combined. As expected, we get a lower variance with a higher level of fusion. For instance, we get a lower variance when the level of fusion is 3 and there are 15 sensors in the system than when there is no fusion and there are 35 sensors in the system. Therefore, by performing fusion, for the same QoI, we can deploy fewer sensors in the system. Further, we can see that as the number of sensors that are allowed to perform fusion increases, the drop in variance decreases. For instance, there is a higher drop in variance between fusion levels 2 and no fusion than between fusion levels 3 and 2. This means that increasing the fusion level beyond a certain amount in this case does not necessarily result in a significant improvement in performance.

We now fix the level of fusion to 3, and vary the number of targets in the system (Figure 10(b)). Clearly, as the number of targets increase, the total variance increases. Notice the difference in variance when there are 10 targets in the system, and when there are 2 targets. While this difference is nearly 3 when there are 5 sensors, it is around 0.75 when there are 25 sensors, and 0.5 when there are 50 sensors. Thus, increasing from 25 to 50 sensors only results in a drop of 0.25. This again illustrates the fact that, depending on the system parameters, it may not be necessary to have a high sensor density.

6. OTHER SENSOR APPLICATIONS

Target tracking is only one of the many applications involving a sensor network. It may even be only one of the many tasks the network performs; a network may simultaneously need to track a target, capture images, determine temperature, detect anomalous signals, etc. Sensors in a network may be used to take temperature readings and localize hot spots such as fires, and chemical readings to localize and predict the movement of hazardous gases. Our framework is applicable to a variety of these sensor applications. Each sensor operation can have its own interpretation of reward, such as the QoI expressed as variance, precision, probability of detection, false alarm, delay, amount of information, relevancy, a combination of these, etc. Once the appropriate reward expression is computed for any combination (fusion) of sensors assigned to the various sensing operations, one can apply our resource allocation framework to maximize the overall reward under energy (or other demand) constraints.

Next we see how to apply our framework to the case of scheduling the flight path of a UAV servicing UGSs.

6.1 UAV Example Application

UAVs are used, among other things, to service sensor nodes on the ground such as collecting information stored in UGSs [5]. The UAV will need to collect as much information as possible under, say, time and/or energy constraints. Let us now consider a specific example in this context. Suppose that sensors (UGSs) are deployed in a field collecting and storing information, and a UAV flies from one sensor to another to collect the stored information. The UAV knows the physical location of the sensors, the time it requires to fly from any sensor to any other sensor, and, say, based on historical data, the expected reward (such as the quality/amount of information) that it can collect from each sensor. The UAV needs to complete its information-collecting task by time δ. The UAV could potentially visit a sensor multiple times within the deadline, if it is known that the sensor has updated information during multiple time slots within the deadline.

Assuming time-slotted operation, we now adapt our framework in Section 3 to this scenario. For each sensor $i \in S$, we associate a collection of sets $\mathbb{T}_i$ representing the possible sets of time slots during which the UAV can visit sensor i. Specifically, $\mathbb{T}_i = \{T_1^i, \ldots, T_{k_i}^i\}$, where for each $l \in \{1, 2, \ldots, k_i\}$, $T_l^i \subseteq \{1, 2, \ldots, \delta\}$. Let $x_{ij} = 1$, if the UAV visits sensor i at each of the time slots in the set $j \in \mathbb{T}_i$, and 0 otherwise. Let t_{ij} be the number of time slots it takes for the UAV to travel from sensor i to sensor j. The problem at hand can now be formulated as follows:

Problem $\Pi_{\mathbf{UAV}}$:

$$\text{maximize} \sum_{i \in S} \sum_{j \in \mathbb{T}_i} q_{ij} x_{ij}$$

$$\text{s.t. (1)} \sum_{j \in \mathbb{T}_i} x_{ij} \leq 1 \text{ for each } i \in S;$$

(2) for any two sensors i_1 and i_2 and for any two sets of slots $j_1 \in \mathbb{T}_{i_1}$ and $j_2 \in \mathbb{T}_{i_2}$, if $\exists k_1 \in j_1$ and $k_2 \in j_2$ s.t. $|k_1 - k_2| < t_{i_1 i_2}$: $x_{i_1 j_1} + x_{i_2 j_2} \leq 1$;

(3) $x_{ij} \in \{0, 1\}$. (15)

The weight of the variable x_{ij} is q_{ij} as shown in the objective function. For this problem, q_{ij} can be any function of the information stored in sensor i, such as its quality, amount, etc., and the set of time slots j that the UAV can visit the sensor. Thus, arbitrary relationships between, say, the QoI and the delay can be modeled within this expression. A very simple model would be to simply consider a ratio, or a weighted difference between the QoI and the delay; in other words, the later the time the UAV can collect the stored information, the lower its quality.

Constraint (1) is a matching constraint. The UAV can collect information from a sensor i in at most one of the sets of time slots in $\mathbb{T}_i$. This is because $\mathbb{T}_i$ consists of all possible sets of slots at which the UAV can visit sensor i before the deadline. Constraint (2) is a topological constraint stating that it is impossible for the UAV to fly from sensor i_1 to i_2 (or, vice versa) within a time interval that is less than a required minimum $t_{i_1 i_2}$.

Clearly, this problem formulation shares similarities with the Traveling Salesman Problem (TSP). However, there is an important difference between the two. With our framework, we can explicitly model the weight obtained by a sensor as an arbitrary function of the information stored by the sensor, and the time at which the UAV reaches the sensor. Further, we allow the weights to be arbitrary when the UAV visits a sensor multiple times. This is not possible in TSP. TSP can only model the total weight as a difference between the total information obtained and the total delay to obtain the information. Thus, our problem is more general than TSP.

The next theorem describes finding a solution Π_{UAV} through a properly constructed graph.

THEOREM 6.1. *An optimal solution to Π_{UAV} can be obtained by finding a maximum weight independent set in the graph $\mathcal{G}'$ constructed as follows:*

1. *for each $i \in S$, and for each $j \in \mathbb{T}_i$, create a node (i, j);*

2. *assign a weight q_{ij} to a node (i, j); and*

3. *create an edge between two nodes* (i_1, j_1) *and* (i_2, j_2) *if and only if one of the following conditions hold:*

   (a) $i_1 = i_2$, *or*

   (b) *there exists* $k_1 \in j_1$ *and* $k_2 \in j_2$ *such that* $|k_1 - k_2| < t_{i_1 i_2}$.

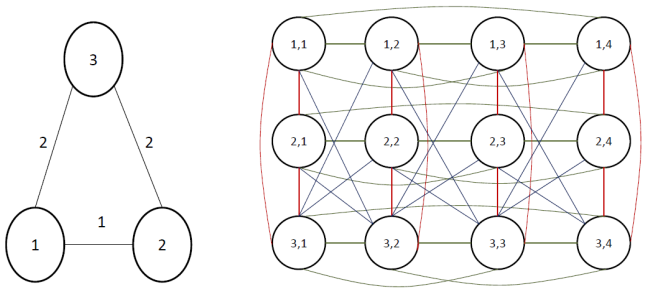
PROOF. The proof is similar to the constructive proof of Theorem 4.1. □

It follows easily that:

COROLLARY 6.1. *A maximum weight independent set in* $\mathcal{G}'$ *explicitly provides the path that the UAV needs to take in order to obtain the optimal solution to* Π_{UAV}.

We could add additional constraints to this problem. One possibility could be that the UAV must collect information from at least k sensors within the deadline. Another possibility is having constraint on energy (e.g., fuel consumption). We can model these constraints as linear constraints and use our Lagrangian approach to solve these problems.

We now provide an example to illustrate this application (Figure 11). Suppose that there are three sensors in the system collecting information. As shown in Figure 11(a), suppose that it takes the UAV one time slot to fly between sensors 1 and 2, and two time slots to fly between sensor 3 and any of the other two sensors. Suppose that $\delta = 4$, and each sensor is visited by the UAV only once within the deadline. Then the graph $\mathcal{G}'$ constructed in Theorem 6.1 is as shown in Figure 11(b). The edges representing the constraint that a sensor can be visited at most once are marked in green, those representing the constraint that at most one sensor can be visited during a single time slot are marked in red, and those representing the topological constraints are marked in blue. It is easy to verify the construction. For instance, there is an edge between $(1, 2)$ and $(3, 3)$ because it is not possible for the UAV to fly from sensor 1 to sensor 3 in one time slot. On the other hand, there is no edge between $(1, 1)$ and $(2, 2)$, or $(1, 1)$ and $(3, 4)$ because it is possible for the UAV to fly from sensor 1 to sensor 2 in one slot, and from sensor 1 to sensor 3 in three slots.



(a) Sensor placement (b) Graph $\mathcal{G}'$

Figure 11: Typical UAV setting

7. RELATED WORK

There is extensive literature related to assignments in target-tracking scenarios which typically assume integer programming formulations. In single target-tracking, [10] studies how to select k out of m total sensors such that the error variance of the combined measurements of the k sensors is minimized. Using convex relaxation techniques, [10] develops a solution for the case that both k and m are large. The proposed solution has limited application when multiple targets are present in the system. In [6], a similar problem is studied for an Extended Kalman Filter where the goal is to choose a group of sensors such that the total energy is minimized subject to constraints on the error variance. Both these studies are computationally taxing as they are complex to solve without relaxing the corresponding integer program, because of the assumption that k is large. However, in practical scenarios, k can be a small number (Figure 10(a)). For example, in order to find the location of a target, one radar or two acoustic sensors are sufficient. Moreover, the sensors whose information are fused will typically be located closer to the target than other sensors. Hence, it is usually not necessary to solve the problem for a high value of k. Also, these studies are limited to specific estimation techniques (such as linear or extended Kalman Filters) and cannot be used for general fusion operations that are considered by our framework.

Regarding multi-target-tracking, [9] shows that this problem is NP-Hard even when information can be fused from only two sensors; it also provides approximation algorithms for the problem by observing a relationship to a bin-packing problem. This study does not consider sensor resource limitations. In [15], this problem is studied under constraints on the reward required for each target, and a solution is obtained by modeling the problem as a semi-matching problem. In both of these studies, a sensor can track only one target, but a target can be tracked by multiple sensors. Also, they assume that the reward obtained from multiple sensors can be expressed as the sum of the rewards obtained from individual sensors. Therefore, while this work takes some constraints on the quality into account, it does not capture general reward metrics as considered in this paper.

Maximum weight independent set techniques have been used for associating sensor measurements to targets in [14, 13]. This is a fundamentally different class of assignment problems than what we considered with our framework, and, for example, operational constraints such as energy are not considered. Hence, we are dealing with graphs whose structure is different from those in these studies.

The use of Lagrangian techniques in our framework has been inspired by the broader integer programming literature, where the addition of constraints is particular challenging. For such problems, using Lagrangian multipliers for the constraints is a powerful tool. In a generalized assignment problem (GAP) [7] with linear budget constraints, using a Lagrange multiplier for the budget constraint results in a dual problem which can be solved using iterative techniques such as the primal-dual sub-gradient algorithm. [7] provides a comprehensive tutorial of this approach. While this approach does not always guarantee an optimal solution for the primal problem, a number of works in integer programming theory study how to obtain an optimal primal solution from the solution that is obtained by the Lagrangian approach [12], [1]. Exploiting similar techniques over a broad set of problems related to sensor network operation, we observed and provided examples of these issues in Sections 4 and 5.

8. CONCLUDING REMARKS

We studied the problem of assigning resources to tasks in a system for performing multiple tasks simultaneously. We developed an integer programming framework for this problem. Compared to existing work, our framework can assign resources to multiple tasks, can be used for maximizing a general function of the reward when information from multiple resources are fused, and satisfies demand constraints. In the absence of demand constraints, we showed that these problems can be solved by finding maximum weight independent sets, many of which have a low computational complexity because of the structure of the graphs in which they were found. We then extended these solutions using a Lagrangian approach when there exist demand constraints. While this approach has been widely used in convex programming, it is of great use in integer programming as well since it can efficiently solve integer programs which become computationally harder when additional constraints are added.

We illustrated a number of applications of our framework for different sensor operations including a problem on how a UAV should traverse a sensor network to gather information from ground sensors under delay constraints. We provided extensive numerical results applying our framework to a real multi-target tracking problem, and gained insightful understandings on the convergence of our algorithms, the performance of the QoI over a time horizon, and the effects of varying the sampling period. All these have profound impact on the efficient (for the network) and beneficial (for its applications) operation and management of the network.

In closing, despite its broadness, we acknowledge that our framework accounts only for the cases that either the reward or the demand functions are arbitrary, but not both of them. The latter cannot be modeled directly using a linear integer programming approach. For TOM, the demand constraint will be non-linear, and for ROM, the objective will be non-linear. Therefore, we cannot utilize structures that occur in linear integer programming problems such as maximum weighted matching, or maximum weight independent set. On the other hand, if the demand function is convex, and the reward function is concave, we can potentially use convex-relaxation techniques to approach this problem. This is an important open problem for future research as is the potential of developing approximate solutions that are built on a hybrid combination of the two models.

9. ACKNOWLEDGEMENTS

The authors would like to acknowledge Dr. Raju Damarla and Dr. Ananthram Swami from ARL for the long and fruitful discussions on the subject of this paper.

10. REFERENCES

[1] A. Berger, V. Bonifaci, F. Grandoni, and G. Schafer. Budgeted matching and budgeted matroid intersection via the gasoline puzzle. *Integer Programming and Combinatorial Optimization, Lecture Notes in Computer Science*, 2008.

[2] D. P. Bertsekas and J. N. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods.* Athena Scientific, 1997.

[3] C. Bisdikian, L. Kaplan, M. Srivastava, D. Thornley, D. Verma, and R. Young. Building principles for a quality of information specification for sensor information. In *12th International Conference on Information Fusion (FUSION '09)*, 2009.

[4] R. Burkard, M. Dell'Amico, and S. Martello. *Assignment Problems.* Society for Industrial and Applied Mathematics, 2009.

[5] J. Burman, J. Hespanha, U. Madhow, D. Klein, J. Isaacs, S. Venkateswaran, and T. Pham. Heterogeneous sensor networks: a bio-inspired overlay architecture. In *Proceedings of SPIE*, 2010.

[6] A. S. Chhetri, D. Morrell, and A. Papandreou-Suppappola. On the use of binary programming for sensor scheduling. *IEEE Transactions on Signal Processing*, 2007.

[7] M. L. Fisher. The Lagrangian relaxation method for solving integer programming problems. *INFORMS Management Science*, 2004.

[8] M. Held, P. Wolfe, and H. P. Crowder. Validation of subgradient optimization. *Mathematical Programming*, 1974.

[9] V. Isler, J. Spletzer, S. Khanna, and T. C. J. Target tracking with distributed sensors: the focus of attention problem. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2003.

[10] S. Joshi and S. Boyd. Sensor selection via convex optimization. *IEEE Transactions on Signal Processing*, 2009.

[11] R. Karp. Reducibility among combinatorial problems. *Complexity of Computer Communications*, 1972.

[12] A. Kulik and H. Shachnai. On Lagrangian relaxation and subset selection problems. *Approximation and Online Algorithms, Lecture Notes in Computer Science*, 2009.

[13] D. J. Papageorgiou and M. R. Salpukas. The maximum weight independent set problem for data association in multiple hypothesis tracking. *Optimization and Cooperative Control Strategies, Lecture Notes in Computer Science*, 2009.

[14] A. B. Poore and A. J. R. III. A new Lagrangian relaxation based algorithm for a class of multidimensional assignment problems. *Computational Optimization and Applications*, 1997.

[15] H. Rowaihy, M. P. Johnson, O. Liu, A. Bar-Noy, T. Brown, and T. La Porta. Sensor mission assignment in wireless sensor networks. *ACM Transactions on Sensor Networks (TOSN)*, 2010.

[16] H. L. van Trees. *Detection, Estimation, and Modulation Theory, Part I.* John Wiley & Sons, 1968.