Efficient Planning of Informative Paths for Multiple Robots

Amarjeet Singh UCLA
Andreas Krause CMU
Carlos Guestrin CMU
William Kaiser UCLA
Maxim Batalin UCLA

Abstract

In many sensing applications, including environmental monitoring, measurement systems must cover a large space with only limited sensing resources. One approach to achieve required sensing coverage is to use robots to convey sensors within this space.Planning the motion of these robots – coordinating their paths in order to maximize the amount of information collected while placing bounds on their resources (e.g., path length or energy capacity) – is a $\mathbf{NP}$-hard problem. In this paper, we present an efficient path planning algorithm that coordinates multiple robots, each having a resource constraint, to maximize the "informativeness" of their visited locations. In particular, we use a Gaussian Process to model the underlying phenomenon, and use the mutual information between the visited locations and remainder of the space to characterize the amount of information collected. We provide strong theoretical approximation guarantees for our algorithm by exploiting the submodularity property of mutual information. In addition, we improve the efficiency of our approach by extending the algorithm using branch and bound and a region-based decomposition of the space. We provide an extensive empirical analysis of our algorithm, comparing with existing heuristics on datasets from several real world sensing applications.

1 Introduction

Mobile robots carrying sensors can enable a large number of real-world, large-scale sensing applications. Consider, for example, the monitoring of algae biomass in a lake. High levels of pollutants, such as nitrates, can lead to the development of algal blooms. These nuisance algal blooms impair the beneficial use of aquatic systems, by blocking sunlight to underwater vegetation, consuming oxygen in the water, and producing surface scum and odors. Precise sensing of quantities, such as pollutants, nutrients, and oxygen levels, can provide biologists with a fundamental characterization of the state of such a lake. Unfortunately, such sensors are a high cost resource, and it is thus impractical to sufficiently cover the lake with these devices.In this setting, a set of robotic boats have been used to move such sensors to various locations in the lake [Dharіwal *et al.*, 2006].

When monitoring algae biomass, or in many other real-world sensing tasks, planning the motion of such robots – coordinating their paths in order to maximize the amount of information collected – is a fundamental task. Often however, such robots have resource constraints, such as storage battery energy, that limit the distance they can travel, or the number of measurements they can acquire.We thus seek to find *informative paths* for a collection of robots, placing a bound on the cost incurred by each robot, e.g., on the battery capacity.

To optimize the path of these robots, we must first characterize the notion of informativeness. Since we are addressing a spatial phenomena, a common approach in spatial statistics is to use a rich class of probabilistic models called *Gaussian Processes* (GPs) (*c.f.*, Rasmussen and Williams 2006). Using such models, informativeness can be viewed in terms of the uncertainty about our prediction of the phenomena, given the measurements made by the mobile robots. In particular, we use the *mutual information* (MI) criterion of Guestrin *et al.* [2005] to quantify the reduction in uncertainty provided by the measurements made along the selected robot paths. Like many other notions of informativeness, mutual information is a *submodular function* [Guestrin *et al.*, 2005], i.e., it satisfies an important diminishing returns property: More the locations that are already sensed, lesser will be the information gained by sensing a new location.

In this paper, we present the first efficient path planning algorithm (e*MIP*) that coordinates multiple robots, each having a resource constraint, in order to obtain highly informative paths, i.e., paths that maximize some given submodular function, such as mutual information. By exploiting the submodularity, we provide strong theoretical approximation guarantees for our algorithm.

The problem of optimizing the path of a *single robot* to maximize a submodular function of the visited locations has been studied by Chekuri and Pal [2005], who provide an algorithm with strong guarantees. We first provide an approach, *sequential-allocation*, for extending *any* single robot algorithm, such as that of Chekuri et al., to the multi-robot setting, with minimal effect on the approximation guarantee.

Unfortunately, the running time of the approach of Chekuri et al. is *quasi-polynomial*, i.e., exponential in $\log_2 M$, for M possible sensing locations, and is thus impractical. Using a spatial decomposition and branch and bound techniques, we develop a practical approach for the single robot case, with theoretical guarantees. Using *sequential-allocation*, we then extend our approach to the multi-robot case. Furthermore, we provide extensive experimental analysis of our algorithm on several real world sensor network data sets, including data collected by the robotic boats in a lake.

2 Problem Statement

Let us now define the *Multi-robot Informative Path Planning* (MIPP) problem: We assume that the domain of the phenomenon is discretized into finitely many sensing locations $\mathcal{V}$. We associate with each location $v \in \mathcal{V}$ a *sensing cost* $C(v) > 0$. When traveling between two locations, u and v, a robot incurs a (not necessarily symmetric) *traveling cost* $C(u, v) > 0$. A robot will traverse a *path* in this space: an s–t-path $\mathcal{P}$ is a sequence of l locations starting at node s, and ending at t. The cost $C(\mathcal{P})$ of path $\mathcal{P} = (s = v_1, v_2, \ldots, v_l = t)$ is the sum of sensing costs and traveling costs along the path, i.e., $C(\mathcal{P}) = \sum_{i=1}^{l} C(v_i) + \sum_{i=2}^{l} C(v_{i-1}, v_i)$. For a collection of k paths $\mathcal{P} = \{\mathcal{P}_1, \ldots, \mathcal{P}_k\}$, one for each robot, let $\mathrm{I}(\mathcal{P}) = \mathrm{I}(\mathcal{P}_1 \cup \cdots \cup \mathcal{P}_k)$ be the *sensing quality*, which quantifies the amount of information collected by the k paths. The MIPP problem desires to find a collection $\mathcal{P}$ of k paths, with specified starting and ending nodes s_i and t_i, such that each path has bounded cost $C(\mathcal{P}_i) \le B$ for some specified budget B, and that the paths are the *most informative*, i.e., $\mathrm{I}(\mathcal{P})$ is as large as possible. Formally, the problem can be defined as:

$$\max_{P_i \subseteq \mathcal{V}} \mathrm{I}(\mathcal{P}); \quad \text{subject to } C(P_i) \le B, \forall\, i \le k. \tag{1}$$

In our lake monitoring example, the single-robot problem instance is depicted in Fig. 1a, top. We try to find the most informative path $\mathcal{P}_1$ (in terms of predicting the algal content). The experiment cost $C(v_i)$ would correspond to the energy required for making a biomass measurement, whereas the travel cost $C(v_{i-1}, v_i)$ would correspond to the energy consumption when traveling from location v_{i-1} to v_i.

Quantifying informativeness. How can we quantify the sensing quality I? To model spatial phenomena, a common approach in spatial statistics is to use a rich class of probabilistic models called *Gaussian Processes* (GPs, *c.f.,* Rasmussen and Williams 2006). Such models associate a random variable $\mathcal{X}_v$ with each location $v \in \mathcal{V}$. The joint distribution $P(\mathcal{X}_\mathcal{V})$ can then be used to quantify uncertainty in the prediction of unobserved locations, after acquiring some measurements. To quantify this uncertainty, we use the *mutual information* (MI) criterion of Guestrin *et al.* [2005]. For a set of locations $\mathcal{P}$, the MI criterion is defined as:

$$\mathrm{MI}(\mathcal{P}) \equiv H(\mathcal{X}_{\mathcal{V}\setminus\mathcal{P}}) - H(\mathcal{X}_{\mathcal{V}\setminus\mathcal{P}} \mid \mathcal{X}_\mathcal{P}), \tag{2}$$

where $H(\mathcal{X}_{\mathcal{V}\setminus\mathcal{P}})$ is the entropy of the unobserved locations $\mathcal{V} \setminus \mathcal{P}$, and $H(\mathcal{X}_{\mathcal{V}\setminus\mathcal{P}} \mid \mathcal{X}_\mathcal{P})$ is the conditional entropy of locations $\mathcal{V} \setminus \mathcal{P}$ after sensing at locations $\mathcal{P}$. Hence mutual information measures the reduction in uncertainty at the unobserved locations. Hence, in our lake monitoring example, we would like to select the locations that most reduce the uncertainty in the algal content prediction for the entire lake.

Even if we do not consider the constraints in the length of the paths of the robots, the problem of selecting locations that maximize mutual information is NP-hard [Guestrin *et al.*, 2005]. Fortunately, mutual information satisfies the following diminishing returns property of Guestrin *et al.* [2005]: The more locations that are already sensed, lesser will be the information gained by sensing a new location. This intuition is formalized by the concept of *submodularity*: A function f is *submodular* if $\forall \mathcal{A} \subseteq \mathcal{B} \subseteq \mathcal{V}$ and $s \in \mathcal{V} \setminus \mathcal{B}$, $f(\mathcal{A} \cup s) - f(\mathcal{A}) \ge f(\mathcal{B} \cup s) - f(\mathcal{B})$. Another intuitive requirement is that the function f is *monotonic*, which means that $f(\mathcal{A}) \le f(\mathcal{B})$ for all $\mathcal{A} \subseteq \mathcal{B} \subseteq \mathcal{V}$. Hence, more the sensing locations that are selected, higher will be the collected information.

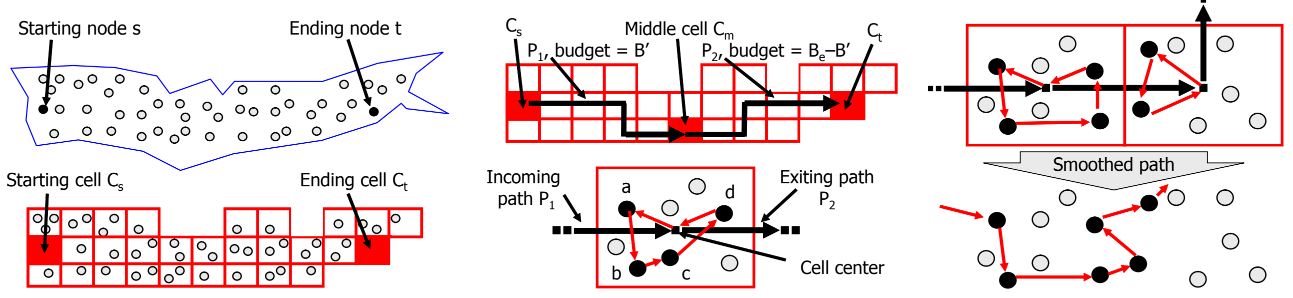
Thus, we define our MIPP problem as one of optimizing paths of length at most B for k robots, such that the selected sensing locations maximize a *monotonic submodular function* $\mathrm{I}(\cdot)$. Note that this definition of the MIPP problem allows our approach to be applied to *any* monotonic submodular objective function, not just mutual information. Guestrin *et al.* [2005] address the sensor placement problem, where a subset $\mathcal{A} \subseteq \mathcal{V}$ of locations are selected in order to maximize the mutual information, without considering path costs. By exploiting the submodularity property of MI, they show that if the discretization $\mathcal{V}$ is fine enough and the GP satisfies mild regularity conditions, greedily selecting locations based on this criterion is near optimal. More specifically, the greedy algorithm (which we call *GreedySubset* in the following), after selecting the first k locations $\mathcal{A}_k$, picks the location $v_{k+1} = \mathrm{argmax}_v\, \mathrm{I}(\mathcal{A}_k \cup \{v\}) - \mathrm{I}(\mathcal{A}_k)$ and sets $\mathcal{A}_{k+1} = \mathcal{A}_k \cup \{v_{k+1}\}$. *GreedySubset* hence iteratively adds locations which increment mutual information the most. Guestrin *et al.* [2005] showed that *GreedySubset* selects sets which achieve mutual information of at least $(1-1/e)\,\mathrm{OPT} - \varepsilon$, where OPT is the optimal mutual information among all sets of the same size, and ε is a small error incurred due to the discretization. This result however only holds true in the unconstrained setting, where k arbitrary locations can be picked, and does not generalize to the MIPP problem. In this paper, we provide an efficient algorithm with strong approximation guarantees for the more difficult MIPP problem.

3 Approximation Algorithm for MIPP

The problem of optimizing the path of a *single robot* (i.e., $k = 1$) to maximize a submodular function of the visited locations has been studied by Chekuri and Pal [2005]. They proposed a *recursive-greedy* algorithm, which provides a $\mathcal{O}(\log |\mathcal{P}^*|)$ approximation guarantee, where $|\mathcal{P}^*|$ is the number of nodes visited in the optimal path $\mathcal{P}^*$, which is no larger than the number of possible locations $|\mathcal{V}|$. That is, their algorithm will provide a path of length no more than B that visits locations yielding a submodular value of at least $\mathcal{O}(\mathrm{OPT} / \log |\mathcal{P}^*|)$, where OPT is the submodular value collected by the optimal path. Their algorithm provides the best approximation guarantee known for the single robot MIPP problem.

The *recursive-greedy* algorithm works by iterating over the possible middle nodes of the path, splitting the path into a left subpath and a right subpath. For each possible middle point, the algorithm is applied recursively on the left subpath. Then, their approach commits to the selected locations on the left side, and recurses on the right subpath, given these selected locations. This algorithm is "greedy" in that it commits to the nodes selected in the first subpath when optimizing the second subpath.

In the case of multiple robots, to our knowledge, no sub-exponential approximation algorithm has been proposed previously. In this paper, we first present an algorithm for the multiple robot setting that exploits any approximation algorithm for the single robot case, such as the *recursive-greedy* algorithm, and (almost) preserves the approximation guarantee. Our algorithm, *sequential-allocation*, works by successively applying the single robot path planning algorithm k times to get the paths for k robots. At stage i, in order not to double-count information from locations already visited in



(a) Spatial decomposition of the phenomenon (b) Cell paths and travel within cells (c) Cell paths and path smoothing

Figure 1: Illustration of e*MIP*. The sensing domain ((a), top) is decomposed into a grid of cells ((a), bottom). e*MIP* jointly optimizes over cell-paths ((b), top) and allocations of experiments in the cells ((b), bottom). Within the cells, nodes are connected to cell center. e*MIP* concatenates the between-cell and within cell paths ((c), top) and finally applies heuristics to smooth the path ((c), bottom).

earlier stages, we supply a modified sensing quality function to the single robot procedure: Let $\mathcal{A}_{i-1}$ be the nodes already visited by paths $\mathcal{P}_1, \ldots, \mathcal{P}_{i-1}$. Then the *residual information*, $\mathrm{I}_{\mathcal{A}_{i-1}}$ is defined as $\mathrm{I}_{\mathcal{A}_{i-1}}(\mathcal{P}) = \mathrm{I}(\mathcal{A}_{i-1} \cup \mathcal{P}) - \mathrm{I}(\mathcal{A}_{i-1})$. This residual information effectively commits to the nodes already visited by the algorithm until stage $i-1$, before deciding the nodes to visit at that stage. The sequential allocation procedure is implemented in Line a1 of Algorithm 1.

Surprisingly, this straight-forward sequential application of the single robot path planning algorithm results in the following approximation guarantee:

Theorem 1. *Let η be the approximation guarantee for the single path instance of the informative path planning problem. Then our sequential-allocation algorithm achieves an approximation guarantee of $(1+\eta)$ for the MIPP problem. In the special case, where all robots have the same starting locations $(s_i = s_j, \forall i,j)$ and finishing locations $(t_i = t_j, \forall i,j)$, the approximation guarantee improves to $1/(1-\exp(-1/\eta)) \leq 1+\eta$.*

All proofs can be found in the longer version of this paper [Singh *et al.*, 2006]. When using the *recursive greedy* algorithm from Chekuri and Pal [2005], the approximation guarantee η is $\mathcal{O}(\log|\mathcal{P}^*|)$ as discussed above. This result extends the analysis of Blum *et al.* [2003], who considered additive functions, to our submodular setting.

4 Efficient Algorithm for MIPP

Unfortunately, the running time of the *recursive-greedy* algorithm is *quasi-polynomial*. More specifically, the running time of the algorithm is $\mathcal{O}((MB)^{O(\log M)})$, where B is the budget constraint and $M = |\mathcal{V}|$ is the total number of nodes in the graph. So, even for a small problem with $M = 64$ nodes, the exponent will be 6, resulting in a very large computation time, making the algorithm impractical for real world sensing applications. In this section, we propose an efficient algorithm for MIPP, *eMIP*, which is based on and has similar approximation guarantees as the *recursive-greedy* algorithm, but is practical for real-world sensing tasks. Exploiting submodularity and using several branch and bound heuristics, we reduce the computation effort to within tractable limits. Our *eMIP* algorithm assumes that the traveling cost between arbitrary locations is given by their Euclidean distance. We discuss the algorithm only for the single robot instance of the problem, since it can be easily extended for multiple robots using Theorem 1.

4.1 Spatial Decomposition

Krause *et al.* [2006] empirically show, that in addition to submodularity, the mutual information criterion exhibits the following *locality* property: Two sets $\mathcal{A}$ and $\mathcal{B}$ of sensing locations which are sufficiently far apart are roughly independent. Hence, in order to obtain a large amount of information, a robot will have to visit several locations that are far from each other, rather than staying in one small area. We can thus think about planning informative paths as deciding which regions to explore, and then deciding which locations to sense in these regions. This motivates the decomposition of the sensing domain into cells, representing clusters of the sensing locations, and then run the *recursive greedy* algorithm on these cells instead of actual sensing locations. Fig. 1 presents an illustration of our approach.

Overview. Informally, our strategy will be the following:

1. We decompose the sensing region (*c.f.,* Fig. 1a, top) into a collection of non-overlapping cells $\widetilde{\mathcal{V}} = \{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_N\}$ (*c.f.,* Fig. 1a, bottom). The distance between two cells is defined as the distance between the centroids of these cells. All nodes $v \in \mathcal{V}$, representing sensing locations, are assigned to the cell $\mathcal{C}_i$ in which they are contained.
2. We define a new optimization problem, the *spatially decomposed MIPP problem, or SD-MIPP problem* on $\widetilde{\mathcal{V}}$. In SD-MIPP, we *jointly* optimize over cell-paths in $\widetilde{\mathcal{V}}$ (*c.f.,* Fig. 1b, top), and over the *allocation* of measurements to the cells visited by the paths. When allocating measurements to a cell, we ignore the traveling cost within the cell (*c.f.,* Fig. 1b, bottom). Since the cells are not excessively large, this simplification only leads to a small additional cost when the SD-MIPP solution is transformed back to original MIPP problem.
3. We transfer the (approximate) SD-MIPP solution, consisting of a cell-path and an allocation of measurements to cells (*c.f.,* Fig. 1c, top), back to the original MIPP problem. We then smooth the path using *tour-opt* heuristics (*c.f.,* Fig. 1c, bottom). The resulting solution will have an approximation guarantee which depends on the diameter of the chosen cells.

More formally, the SD-MIPP problem is the following: Suppose, the budget $\widetilde{B}$ is split into a budget B_t for traveling between the cells, and a budget B_e for making experiments at sensing locations within the visited cells. We want to find a path $\mathcal{P}^* = (\mathcal{C}_s = \mathcal{C}_{i_1}, \ldots, \mathcal{C}_{i_l} = \mathcal{C}_t)$ with a travel cost of at most B_t. This travel budget is measured in terms of distances between centers of visited cells, and the cost of traveling *within* cells is defined as 0. In addition, for each visited cell $\mathcal{C}_{i_j}$ in $\mathcal{P}^*$, we want to select a set of sensing locations $\mathcal{A}_j$, such that the total cost $C(\mathcal{A}_1 \cup \cdots \cup \mathcal{A}_l) \leq B_e$, and that the information $\mathrm{I}(\mathcal{A}_1 \cup \cdots \cup \mathcal{A}_l)$ is as large as possible. The optimal SD-MIPP solution $\mathcal{P}^*$ uses the optimal split of the budget $\widetilde{\mathcal{B}}$ into B_t and B_e. To simplify the presentation, we rescale the costs such that the cells form a uniform grid of quadratic cells with width L, and assume that the sensing cost C_{exp} is constant over all locations. These assumptions can easily be relaxed, but they allow us to relate the path costs to the number of cells traversed, to simplify the discussion.

Algorithm for SD-MIPP. We now present our algorithm for solving the optimization problem on $\widetilde{\mathcal{V}}$. In Section 4.2, we will discuss several details of efficient implementation. The complete algorithm works as follows: An outer loop (*c.f.,* Line b1 in Algorithm 1) iterates through $B_t \in \{2^0, 2^1, 2^2, \ldots, \widetilde{\mathcal{B}}\}$, where $\widetilde{\mathcal{B}}$ is the budget given to the SD-MIPP problem, allocating budget B_t out of the total budget $\widetilde{\mathcal{B}}$ for traveling between the cells, and $B_e = \widetilde{\mathcal{B}} - B_t$ for making experiments within the visited cells. Stepping through the travel budget B_t in powers of 2 results in faster performance ($\log_2 \widetilde{\mathcal{B}}$ instead of $\widetilde{\mathcal{B}}$ iterations), and increases the required budget $\widetilde{\mathcal{B}}$ by at most a factor of 2. The inner loop is formed by a recursive procedure, shown in Algorithm 2, which selects cells to visit, and allocates experiments to cells.

More specifically, this *recursive*-e*MIP* procedure takes as input a starting cell $\mathcal{C}_s$, a finishing cell $\mathcal{C}_t$, an experimental budget B_e, a residual $\mathcal{X}$ indicating the locations visited thus far (initially empty), and a maximum recursion depth, $iter$, which is initialized to $\log_2 B_t$. We then:

1. Iterate through all possible choices of middle cells $\mathcal{C}_m$, and budget splits $B' \in \widetilde{\mathcal{B}_e}$ to spend for making experiments on the path from $\mathcal{C}_s$ to $\mathcal{C}_m$ (*c.f.,* Fig. 1b). The budget splits $\widetilde{\mathcal{B}_e}$ can either be linearly (more accurate) or exponentially (faster) spaced, as described below.
2. Recursively find a path $\mathcal{P}_1$ from $\mathcal{C}_s$ to $\mathcal{C}_m$, subtracting 1 from the maximum recursion depth, $iter$. This maximum recursion depth controls the maximum number of cells visited by $\mathcal{P}_1$. At the top level of the recursion, $\mathcal{P}_1$ will visit a maximum of $B_t/2$ cells, in the next level, the limit will be $B_t/4$ cells, and so on. When reaching a maximum recursion depth of 0, we use the *Greedy-Subset* algorithm (*c.f.,* Section 2) to select the sensing locations $\mathcal{A}_i$ based on the residual information function $\mathrm{I}_{\mathcal{X}}$ constrained by budget B'. Hereby, the residual $\mathcal{X}$ is a parameter of the recursion, and contains all nodes visited before considering the current cell. As an illustration, consider the black nodes selected in the middle cell $\mathcal{C}_m$ in Fig. 1b, bottom. These have been selected by the *GreedySubset* algorithm with budget $B' = 4$, since they provide the maximum improvement in mutual information measured against the path $\mathcal{P}_1$ of Fig. 1b, top.

```
   Algorithm: eMIP
   Input: $B$, $k$, starting / finishing nodes $s_1, \ldots, s_k, t_1, \ldots, t_k$
   Output: A collection of informative paths $\mathcal{P}_1, \ldots, \mathcal{P}_k$.
   begin
       Perform spatial decomposition into cells;
       Find starting and ending cells $\mathcal{C}_{s_i}$ and $\mathcal{C}_{t_i}$;
       $X \leftarrow \emptyset$;
a1     for $i = 1$ to $k$ do
b1         for $iter = 0$ to $\lfloor \log_2 B \rfloor$ do
               $B_e \leftarrow B - 2^{iter}$;
               $\mathcal{P}'_{iter} \leftarrow$ recursive-eMIP($\mathcal{C}_{s_i}$,$\mathcal{C}_{t_i}$,$B_e$,$X$,$iter$);
               Smooth $\mathcal{P}'_{iter}$ using tour-opt heuristics;
           $\mathcal{P}_i \leftarrow \mathrm{argmax}_{iter}\, \mathrm{I}(\mathcal{P}'_{iter})$;
           $X \leftarrow X \cup \mathcal{P}_i$;
       return $\mathcal{P}_1, \ldots, \mathcal{P}_k$;
   end
```

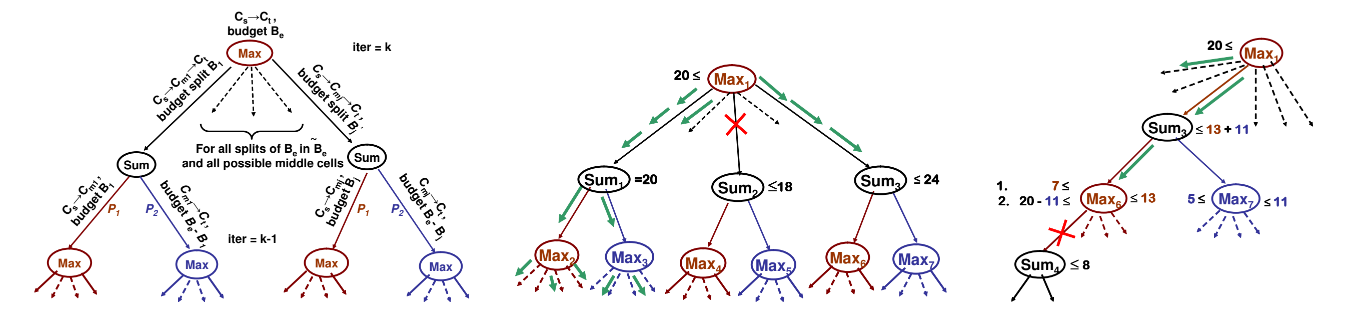
Algorithm 1: e*MIP* algorithm for informative multiple path planning, realizing the *sequential allocation* described in Section 3 (*c.f.,* Line a1). The path for the i-th robot is found using the spatial decomposition approach described in Section 4, which calls the recursive procedure (*c.f.,* Algorithm 2).

3. Commit to the nodes visited in $\mathcal{P}_1$, and recursively find a path $\mathcal{P}_2$ from $\mathcal{C}_m$ to $\mathcal{C}_t$, with experimental budget $B_e - B'$. This path will also visit at most $B_t/2$ cells. We again greedily select the sensing locations at maximum recursion depth of 0, but now based on the residual information $\mathrm{I}_{\mathcal{X} \cup \mathcal{P}_1}$, since we have committed to $\mathcal{P}_1$.
4. Concatenate the nodes obtained in $\mathcal{P}_1$ and $\mathcal{P}_2$ to output the best path from the algorithm (*c.f.,* Fig. 1c, top).

The *recursive*-e*MIP* procedure is based on the recursive greedy algorithm of Chekuri and Pal [2005], but exploits our spatial decomposition.

Linear vs. exponential budget splits. Step 1 considers different budget splits $B' \in \widetilde{\mathcal{B}_e}$ to the left and right subpaths. Similar to the recursive greedy algorithm, one can choose $\widetilde{\mathcal{B}_e} = \{0, 1, 2, 3, \ldots, B_e - 1, B_e\}$ to be linearly spaced. Since the branching factor is proportional to the number of considered splits, linear budget splits lead to a large amount of computation. An alternative is to consider only exponential splits: $\widetilde{\mathcal{B}_e} = \{0, 1, 2, 4, \ldots, B_e\} \cup \{B_e, B_e{-}1, B_e{-}2, B_e{-}4, \ldots, 0\}$. Here the branching factor is only logarithmic in the experimental budget. Even though we are not guaranteed to find the same solutions as with linear budget splits, we can both theoretically and empirically show that the performance only gets slightly worse in this case, compared to a dramatic improvement in running time. In addition to these two ways of splitting the budget, we also considered one-sided exponential budget splits (i.e., $\widetilde{\mathcal{B}_e} = \{0, 1, 2, 4, \ldots, B_e\}$), which halves the branching factor compared to the exponential splits defined above. Although we do not provide theoretical guarantees for this third possibility, we experimentally found it to perform very well (*c.f.,* Section 5).

Solving the MIPP problem. Now we need to transfer the approximately optimal solution obtained for SD-MIPP back to MIPP. This is done by connecting all nodes selected in cell $\mathcal{C}_i$ to the cell's center, (as indicated in Fig. 1b bottom), then connecting all selected centers to a path (Fig. 1c top), and finally expanding the resulting tree into a tour by traversing



(a) *sum-max* tree (b) Pruning of *sum* nodes (c) Tighter lower bounds

Figure 2: Illustration of our branch & bound approach. (a) shows the *sum-max* tree representing the search space. Each *max* node selects a middle cell and a budget allocation, and each *sum* node combines two subpaths. (b) shows how upper bounds at *sum* nodes are used to prune branches. (c) shows how lower bounds at *max* nodes are tightened to allow even more pruning.

```
Algorithm: recursive-eMIP
Input: $\mathcal{C}_s, \mathcal{C}_t, B_e, X, iter$
Output: An informative path $\mathcal{P}$ from $\mathcal{C}_s$ to $\mathcal{C}_t$.
begin
    if $(d(C_s, C_t) > 2^{iter} L)$ then return Infeasible;
a2  $\mathcal{P} \leftarrow GreedySubset_{B_e}(v_i : v_i \in \mathcal{C}_s \cup \mathcal{C}_t)$;
    if $(iter = 0)$ then return $\mathcal{P}$;
    $reward \leftarrow \mathrm{I}_X(\mathcal{P})$;
b2  foreach $\mathcal{C}_m \in \mathcal{C}$ do
c2      for $B' \in \widetilde{\mathcal{B}_e}$ do
            $\mathcal{P}_1 \leftarrow$ recursive-eMIP$(\mathcal{C}_s, \mathcal{C}_m, B', X, iter - 1)$;
            $\mathcal{P}_2 \leftarrow$ recursive-eMIP$(\mathcal{C}_m, \mathcal{C}_t, B_e - B', X \cup \mathcal{P}_1, iter - 1)$;
            if $(\mathrm{I}_X(\mathcal{P}_1.\mathcal{P}_2) > reward)$ then
                $\mathcal{P} \leftarrow \mathcal{P}_1.\mathcal{P}_2$;
                $reward \leftarrow \mathrm{I}_X(\mathcal{P})$;
    return $\mathcal{P}$;
end
```

Algorithm 2: *recursive*-e*MIP* procedure for path planning.

the tree twice. This traversal results in a tour which is at most twice as long as the shortest tour connecting the selected vertices. (Of course, an even better solution can be obtained by applying an improved approximation algorithm for TSP, such as Christofide's algorithm [1976].) The following Theorem completes the analysis of our algorithm:

Theorem 2. *Let $\mathcal{P}^*$ be the optimal solution for the MIPP problem with a budget of B. Then, our* eMIP *algorithm will find a solution $\widehat{\mathcal{P}}$ achieving an information value of at least $\mathrm{I}(\widehat{\mathcal{P}}) \geq \frac{1-1/e}{1+\log_2 N}\mathrm{I}(\mathcal{P}^*)$, whose cost is no more than $2(2\sqrt{2}B+4L)(1+L\frac{\sqrt{2}}{C_{exp}})$ in the case of linear budget splits (for $\widetilde{\mathcal{B}_e}$) and no more than $2(2\sqrt{2}B+4L)(1+L\frac{\sqrt{2}}{C_{exp}})N^{\log_2 \frac{3}{2}}$ in the case of exponential budget splits (for $\widetilde{\mathcal{B}_e}$).*

Running time analysis of e*MIP* is straightforward. The algorithm calls the routine *recursive*-e*MIP* $\log_2 B$ times. If T_I is the time to evaluate the mutual information I, then time for computing greedy subset T_{gs} (Line *a2*) is $\mathcal{O}(N_C^2\ T_\mathrm{I})$, where N_C is the maximum number of nodes per cell. At each recursion step we try all the cells that can be reached with the available traveling budget (Line *b2*). For the possible experimental budget split, we try all (linearly or exponentially spaced) splits of $B_e \in \widetilde{\mathcal{B}_e}$ among the two paths $\mathcal{P}_1$ and $\mathcal{P}_2$ (Line *c2*). The recursion depth would be $\log_2(\min(N, \widetilde{\mathcal{B}}))$. The following proposition states the running time for e*MIP*.

Proposition 3. *The worst case running time of* eMIP *for linearly spaced splits of the experimental budget is $\mathcal{O}\left(T_{gs}\log_2 B(NB)^{\log_2 N}\right)$, while for the exponentially spaced splits of the experimental budget it is $\mathcal{O}\left(T_{gs}\log_2 B(2N\log_2 B)^{\log_2 N}\right)$*

Comparing this running time to that of the original Chekuri et al. algorithm ($\mathcal{O}((MB)^{O(\log_2 M)})$), we note a reduction of B to $\log_2 B$ in the base, and log of the number of nodes ($\log_2 M$) to log of the number of cells ($\log_2 N$) in the exponent. These two improvements turned the impractical recursive-greedy approach into a much more viable algorithm.

4.2 Branch and Bound

The spatial decomposition technique effectively enables a trade-off between running time complexity and achieved approximation guarantee. However, the e*MIP* algorithm still has to solve a super-polynomial, albeit sub-exponential, search problem.

The problem structure can be represented by a *sum-max* tree as shown in Fig. 2a. The *sum* nodes correspond to combining the two paths $\mathcal{P}_1$ and $\mathcal{P}_2$ on either side of the middle cell $\mathcal{C}_{m_i}$. The *max* nodes correspond to selecting the best possible path for all possible experimental budget split in $\widetilde{\mathcal{B}_e}$ and all possible middle cells $\mathcal{C}_{m_i}$. Thus, each *sum* node will have two children, each one a *max* node, representing the best possible solution from paths $\mathcal{P}_1$ and $\mathcal{P}_2$ respectively. The root of the tree will be a *max* node selecting the best possible solution at the highest level. The depth of the tree, *iter*, depends on the traveling budget B_t, $iter = \lceil\log_2 B_t\rceil$. As an initial pruning step, we note that since the middle cells have to lie on a path, we only need to consider those which are at most distance $B_t/2 \cdot L$ from the starting and finishing cell.

In order to avoid exploring the entire *sum-max* tree and waste computation considering possibly bad solutions, we follow a branch and bound approach. At each *max* node we derive upper bounds for all the child *sum* nodes. We can then prune all the *sum* node children with upper bounds lower than the current best solution. If any of the child *sum* nodes provides a solution better than the current best solution, then the current best solution for the parent *max* node is updated with the improved solution. Fig. 2b presents an illustration of this concept: After completely exploring branch Sum_1, the current best solution, 20, is thus a lower bound for Max_1. The upper bound for branch Sum_2 is 18, which is lower than the current best solution of 20, and hence we can prune this branch and need not explore it further. Nodes such as Sum_3 however, which have higher upper bounds (24) than the current best solution (20), need to be explored further. In order to improve the current best solution faster, at each *max* node we explore the *sum* nodes in the decreasing order of their upper bounds. (An additional heuristic that is very effective in practice is to explore only the top K *sum* nodes.)

Upper bound on the *sum* nodes. One approach for acquiring upper bounds on the *sum* nodes is to relax the path constraints, and then find the optimal set of reachable nodes for each path ($\mathcal{P}_1$ and $\mathcal{P}_2$). In order to compute upper bounds for $\mathcal{P}_1$ ($\mathcal{P}_2$) and their corresponding *max* nodes, we hence need to compute the best *set* of observations within all reachable nodes. Since this problem itself is NP-hard, we approximate it using the *GreedySubset* algorithm: For an allocation of k experiments to $\mathcal{P}_1$ ($\mathcal{P}_2$), we run the *GreedySubset* algorithm to select k nodes reachable (w.r.t. the remaining traveling budget) within this path. Since this algorithm guarantees a constant factor $(1-1/e)$ approximation [Nemhauser *et al.*, 1978], multiplying the resulting information value by $(1-1/e)^{-1}$ provides an upper bound on the information achievable by the path (and hence the corresponding *max* node). In Fig. 2c for example, we use the greedy algorithm to get upper bounds 13 for Max_6 and 11 for Max_7, resulting in an upper bound of $13+11=24$ for Sum_3. We can even compute tighter online bounds for maximizing monotonic submodular functions, as discussed by Nemhauser *et al.* [1978].

Lower bound on the *max* nodes. In order to perform pruning on lower levels of the *sum-max* tree, we need lower bounds for the *max* nodes. Instead of having to explore one branch completely as described above, we have two ways for acquiring such lower bounds: Based on a heuristic algorithm, and based on the current best solution of the grandparent *max* node. We use the larger of two different lower bounds.

The first lower bound is based on a heuristic proposed for the (modular) orienteering problem in [Chao *et al.*, 1996]. The solution obtained by the heuristic immediately provides a lower bound at each *max* node. The heuristic is applied to the actual sensing locations $\mathcal{V}$, instead of the cells. Starting node and finishing node for the heuristic are selected greedily from the starting cell and the finishing cell respectively. The current traveling budget is added to the available experimental budget to calculate the budget constraint for the heuristic.

A second pruning bound for the *max* nodes is given by the difference between the lower bound (the current best solution) of the grandparent *max* node, and the upper bound on the other path originating from the parent *sum* node. This is illustrated in Fig. 2c: For the node Max_6 and Max_7, a lower bound of 7 and 5 respectively is calculated using the heuristic. The current lower bound at the grandparent *max* node (Max_1) is 20. The parent *sum* node is explored further since the sum (24) of the upper bounds from each of the child *max* nodes (13 and 11 for $\mathcal{P}_1$ (Max_6) and $\mathcal{P}_2$ (Max_7) respectively) is greater than the current best solution (20) of the parent *max* node. A potentially tighter pruning bound is calculated by subtracting the upper bound of $\mathcal{P}_2$ ($\mathrm{Max}_7 \leq 11$) from the current best solution of the grandparent *max* node ($\mathrm{Max}_1 \geq 20$). This bound potentially allows to prune branches since, given the upper bound on the second path, improving on the current solution of the grandparent requires exceeding this pruning bound. In our example, this pruning bound (9) is tighter than the lower bound provided by the heuristic (7), and hence enabled pruning of branch Sum_4 (with upper bound 8). This pruning would not have been possible when only using the lower bound calculated using the heuristic (7).

Before exploring the second path $\mathcal{P}_2$, the exact reward collected by the first path $\mathcal{P}_1$ can be used for calculating this additional lower bound. When exploring $\mathcal{P}_1$ however, the exact reward for $\mathcal{P}_2$ is not known, hence only the upper bound calculated using the greedy algorithm as described above can be used. In order to ensure that this lower bound is as tight as possible, we first explore the child of the *sum* node having more budget in the current budget split instead of always exploring $\mathcal{P}_1$, the path from starting cell to middle cell first.

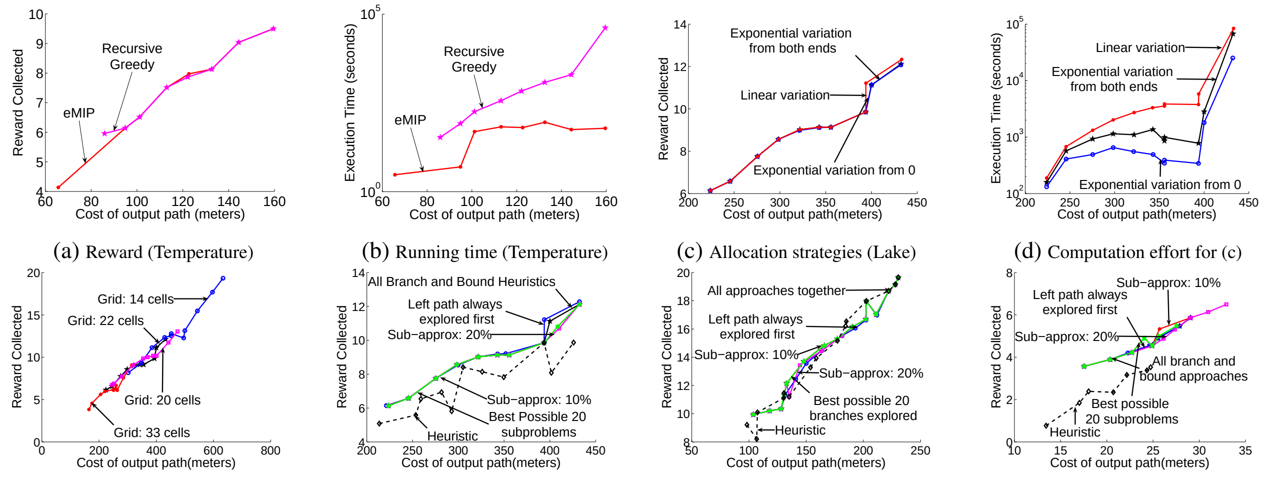
Sub-approximation. Lower and upper bounds can be quite loose. We can address this issue, and further trade off collected information with improved execution time, by introducing a sub-approximation heuristic: instead of comparing the lower bound of a *max* node directly with the upper bound from the children's *sum* nodes when deciding which subproblems to prune, we scale up the lower bound by a factor of $\alpha > 1$. This scaling often allows us to prune many branches that would not be pruned otherwise. Unfortunately, this optimistic pruning can also cause use to prune branches that should not be pruned, and decrease the information collected by the algorithm. Fortunately, we can prove that this approach will not decrease the quality of the solution by more than $1/\alpha$. Furthermore, in practice, for sufficiently small α values, this procedure can speed up the algorithm significantly, without much effect on the quality of the solution.

5 Experiments and Results

5.1 Data Sets

In order to evaluate the performance of our algorithm, we tested it on three real world datasets. Our main set of experiments are on measurements of the biomass content in Lake Fulmor, James Reserve [Dhariwal *et al.*, 2006]. We used data collected by a boat carrying a temperature sensor around the lake, of width around 50 meters and length around 250 meters. Temperature was previously found to be strongly correlated with the algal bloom in the lake. The average speed of the boat was approximately 0.4 m/s. Half of the total measurements (218 different sensing locations) were used to learn a nonstationary Gaussian Process model by maximizing the marginal likelihood [Rasmussen and Williams, 2006], and the rest of the measurements were used for experimentation. We divided the lake into 22 cells, with distance between adjacent cell approximately 21 meters. Based on the average speed, and motivated by a typical measurement duration of roughly 25 seconds, we set the experiment cost to be 10.5 meters.

As our second dataset, we used the existing deployment of 52 wireless sensor motes to learn the amount of temperature variability at Intel Research Berkeley. We divided the



(a) Reward (Temperature) (b) Running time (Temperature) (c) Allocation strategies (Lake) (d) Computation effort for (c)

(e) Cell density (Lake) (f) Branch & bound (Lake) (g) Branch & bound (Temp.) (h) Branch & bound (Precipit.)

Figure 3: Single robot experiments on three real-world data sets. Note the logarithmic scale on the running time plots (Figures 3b and 3d).

complete region into a uniform grid containing 20 equal sized cells, and determined the experimental cost to be 9m (distance to travel between adjacent cells). We learned a GP model as discussed by Krause *et al.* [2006].

Thirdly, we explored the performance of our algorithm on a precipitation dataset collected from 167 regions during the years 1949-1994. We followed the preprocessing and model learning described by Guestrin *et al.* [2005].

5.2 Single Robot Experiments

We first analyze the performance improvement of our e*MIP* algorithm compared to the *recursive-greedy* procedure [Chekuri and Pal, 2005]. Fig. 3a presents trade-off curves comparing path costs and collected reward, for different budget values B on a subset of temperature data set containing only 23 nodes as *recursive-greedy* was intractable on larger datasets. Fig. 3b presents the corresponding running times. We can see that the e*MIP* algorithm achieves (almost) the same amount of mutual information as *recursive-greedy*, but at several orders of magnitude lower running time. Since the recursive greedy algorithm is essentially a search procedure with greedily restricted search space, this result also indicates that an exhaustive search over all paths is intractable.

We then compare the impact of restricting ourselves to the exponentially spaced experimental budget allocation. Figures 3c and 3d present the results on the lake data set. As expected, linear variation achieves very slightly larger collected mutual information than the exponential variation. However, the computation times for exponential variation are several orders of magnitude smaller than for linear variation.

Our next experiment considers the effect of varying the coarseness of spatial decomposition. Fig. 3e shows the results of this experiment on the lake data, indicating that the mutual information is largely insensitive to the coarseness of the cells. On the other hand, the computational complexity decreases drastically as fewer cells are used (*c.f.,* Proposition 3).

We performed the same experiments on both the temperature and precipitation data. Detailed results are omitted here due to space limitations, but they confirm the above insights. In order to analyze the impact of several branch and bound heuristics, we plotted trade-off curves for varying budgets, for the lake (Fig. 3f), temperature (Fig. 3g) and precipitation data (Fig. 3h). We find that the mutual information collected is rather insensitive to sub-approximation (up to 20%), as well as to the restriction to 20 best sub-problems (exploring only the 20 best *sum* nodes per branch). We also compared e*MIP* to a heuristic search algorithm [Chao *et al.*, 1996]. This heuristic had been empirically found to be one of the best heuristics for the similar problem with fixed reward on each node [Liang *et al.*, 2002], so we would expect it to also perform well in the submodular case. We can see that, while for the smaller temperature data set, the heuristic achieves comparable performance, for both larger data sets (lake and precipitation), e*MIP* strongly outperforms the heuristic.

5.3 Multiple Robot Experiments

Fig. 4a shows the comparison of collected reward when multiple robots are available to move the sensors around. As the number of robots are increased, the collected reward exhibits the expected diminishing return, due to the submodularity of mutual information. Fig. 4b presents the same analysis when the robots can start at different locations. Here, four locations, at four corners of the lake, were pre-selected as possible starting locations. The three robots greedily selected the starting location leading to the largest increase in mutual information.

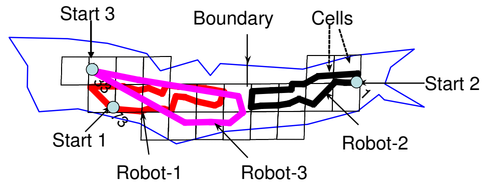
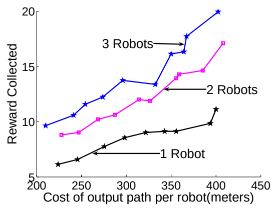
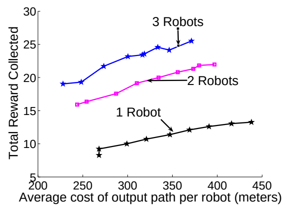


Figure 5: Paths on the lake for different starting cells.

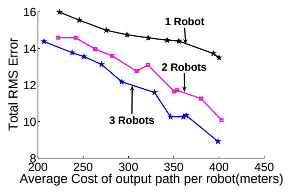
As expected, the first two robots choose starting locations on opposite ends of the lake and collect roughly independent information. With addition of the third robot, the diminishing returns in collected reward can again be observed. Figures 4c and 4d show the predictive RMS error for this exper-



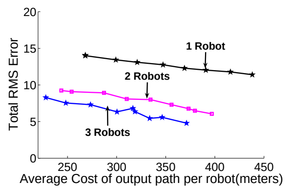
(a) Multi-robot: reward, same start



(b) Multi-robot: reward, different start



(c) Multi-robot: RMS, same start



(d) Multi-robot: RMS, different start

Figure 4: Experiments using multiple robots on the lake dataset.

iment. Analogously to the information value, the RMS error decreases more quickly if the three robots start at different locations, and the biggest improvement as as expected is in the step from one to two robots. Fig. 5 shows the chosen paths for the case of three robots, each starting from different locations.

6 Related Work

A related problem to MIPP is one where each node has a fixed reward, and the goal is to find a path that maximizes the sum of these rewards (*Traveling Salesman Problem with Profits* (TSPP) [Feillet *et al.*, 05]). Such sum is a *modular* function, rather than the submodular function addressed in this paper. A subcategory of TSPP, the *Orienteering Problem* (OP) is defined to maximize the collected reward while keeping the associated cost is defined less than given budget B [Laporte and Martello, 1990]. Multi-robot version of OP is studied in literature as the *Team Orienteering Problem* [I-Ming *et al.*, 1996]. For the case of unrooted version of OP (when no starting location is specified), the approximation guarantees known for *Prize Collecting TSP* and k-TSP can be easily extended [Johnson *et al.*, 2000]. Blum *et al.* [2003] gave the first constant factor approximation for the rooted OP in general undirected graphs. They also extended their algorithm for Multi-path OP. For OP with submodular reward fuction, the approach of Chekuri and Pal [2005], discussed in Section 3, provides the best approximation guarantee, but has a quasi-polynomial running time. In the robotics community, similar work has been developed in the context of *simultaneous localization and mapping* (SLAM). Stachniss *et al.* [2005] develop a greedy algorithm, without approximation guarantees, for selecting the next location to visit to maximize information gain about the map. In contrast, Sim and Roy [2005] attempt to optimize the entire trajectory, not just the next step, but their algorithm introduces some approximation steps without theoretical bounds. We also expect our approach to be useful in the SLAM setting.

7 Conclusions

In this paper, we presented the first efficient path planning algorithm that coordinates multiple robots, each having a resource constraint, in order to obtain highly informative paths, i.e., paths that maximize some given submodular function, such as mutual information. We first described *sequential-allocation*, an approach for extending *any* single robot algorithm to the multi-robot setting, with minimal effect on the approximation guarantee. Then, building on the, impractical, single robot approach of Chekuri and Pal [2005], we developed e*MIP*, a practical algorithm for obtaining an informative path for a single robot with theoretical guarantees, that exploits spatial decomposition and branch and bound techniques. Using *sequential-allocation*, we then extended our approach to the multi-robot case. Furthermore, we provided extensive experimental analysis of our algorithm on several real world sensor network data sets, including data collected by robotic boats in a lake, demonstrating the effectiveness and practicality of our methods.

Acknowledgements This work was supported by NSF Grant Nos. CNS-0509383, CNS-0625518, ANI-00331481 and a gift from Intel Corporation. Carlos Guestrin was partly supported by an Alfred P. Sloan Fellowship and an IBM Faculty Fellowship. We would like to thank Bin Zhang for providing the lake data set.

References

Avrim Blum, Shuchi Chawla, David R. Karger, Terran Lane, Adam Meyerson, and Maria Minkoff. Approximation algorithms for orienteering and discounted-reward tsp. In *FOCS*, page 46, 2003.

I-Ming Chao, Bruce L. Golden, and Edward A. Wasil. A fast and effective heuristic for the orienteering problem. *Eur. Jour. of Op. Research*, 88:475–489, 1996.

Chandra Chekuri and Martin Pal. A recursive greedy algorithm for walks in directed graphs. In *FOCS*, pages 245–253, 2005.

N. Christofides. Worst-case analysis of a new heuristic for the traveling salesman problem. *Tech report,CMU*, 1976.

Amit Dhariwal, Bin Zhang, Beth Stauffer, Carl Oberg, Gaurav S. Sukhatme, David A. Caron, and Aristides A. Requicha. Networked aquatic microbial observing system. In *IEEE ICRA*, 2006.

Dominique Feillet, Pierre Dejax, and Michel Gendreau. Traveling salesman problem with profits. *Trans Sci*, 39(2):188–205, '05.

Carlos Guestrin, Andreas Krause, and Ajit Paul Singh. Near-optimal sensor placements in gaussian processes. In *ICML*, 2005.

C. I-Ming, B.L. Golden, and E.A. Wasil. The team orienteering problem. *Eur. Jour. of Op. Res.*, 88:464–474, 1996.

David S. Johnson, Maria Minkoff, and Steven Phillips. The prize collecting steiner tree problem: theory and practice. In *Symp. on Disc. Algorithms*, pages 760–769, 2000.

Andreas Krause, Carlos Guestrin, Anupam Gupta, and Jon Kleinberg. Near-optimal sensor placements: Maximizing information while minimizing communication cost. In *IPSN*, 2006.

Gilbert Laporte and Silvano Martello. The selective travelling salesman problem. *Disc. App. Math*, 26:193–207, 1990.

Yun-Chia Liang, Sadan Kulturel-Konak, and Alice E. Smith. Meta heuristics for the orienteering problem. In *Proc. of CEC*, 2002.

G. Nemhauser, L. Wolsey, and M. Fisher. An analysis of the approximations for maximizing submodular set functions. *Mathematical Programming*, 14:265–294, 1978.

Carl Edward Rasmussen and Christopher K.I. Williams. *Gaussian Process for Machine Learning*. Adaptive Computation and Machine Learning. MIT Press, 2006.

R. Sim and N. Roy. Global a-optimal robot exploration in slam. In *ICRA*, 2005.

Amarjeet Singh, Andreas Krause, Carlos Guestrin, William Kaiser, and Maxim Batalin. Efficient planning of informative paths for multiple robots. Technical Report CMU-ML-06-112, 2006.

C. Stachniss, G. Grisetti, and W. Burgard. Information gain-based exploration using rao-blackwellized particle filters. In *RSS*, 2005.