Nonmyopic Informative Path Planning in Spatio-Temporal Models

Alexandra Meliou
Andreas Krause
Carlos Guestrin
Joseph M. Hellerstein

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2007-44
http://www.eecs.berkeley.edu/Pubs/TechRpts/2007/EECS-2007-44.html

April 18, 2007

Nonmyopic Informative Path Planning in Spatio-Temporal Models

Alexandra Meliou
UC Berkeley

Andreas Krause
CMU

Carlos Guestrin
CMU

Joseph Hellerstein
UC Berkeley

Abstract

In many sensing applications, like environmental monitoring, we must continuously gather information in order to provide a good estimate of the state of the environment at every point in time. A robot may tour an environment, gathering information every hour. In a wireless sensor network, these tours correspond to packets being transmitted. In these settings, we are often faced with resource restrictions, like energy constraints. When users issue queries, they have certain expectations on the answer quality. Thus, we must optimize the tours in order to ensure the satisfaction of the user constraints, while at the same time minimize the cost of the query plan. For a single timestep, this optimization problem is NP-hard, but recent approximation algorithms with theoretical guarantees provide good solutions. In this paper, we present a new efficient algorithm, exploiting dynamic programming and submodularity of the information being collected, that efficiently plans data collection tours for an entire (finite) horizon. Our algorithm can use any single step procedure as a black box, and provides strong theoretical guarantees about the solution, based on the properties of the single step approach. We also provide an extensive empirical analysis demonstrating the benefits of nonmyopic planning in two real world sensing applications.

Introduction

Many practical applications (e.g., building automation, environmental control, reconnaissance) require the monitoring of various physical phenomena that change over time. The sensing devices used in such applications can often be hard to recharge, repair or replace, posing limitations to their use. These resource constraints demand nuanced schemes for collecting observations that tolerate bounded uncertainty in exchange for reduced resource consumption. In wireless sensor networks for example, sensors have limited battery life. In order to conserve power, one can use data gathering tours (Meliou *et al.* 2006) to acquire measurements at minimum cost. In robotic applications, there can be limits on fuel. Hence one has to plan robot trajectories in order to efficiently acquire information (*c.f.,* Singh *et al.* 2007). Common to these problems is the need to find *maximally informative paths*, while *minimizing the traversal cost* incurred.

In the case of monitoring with sensor networks, the requirements for *informativeness* are usually determined by a user, who often specifies desired confidence requirements on the returned result. This problem is the focus of earlier work (Deshpande *et al.* 2004), where model-based querying is proposed as a new approach to approximate query answering. In order to assess the informativeness of observations before acquiring the actual values, one can use probabilistic models. When dealing with spatial phenomena, as in the current common uses of sensor networks, *Gaussian Processes* (*c.f.,* Rasmussen & Williams 2006) have been successfully used as such models. These models allow us to quantify the informativeness of selected observations in terms of the expected reduction in predictive variance.

Most existing work in the area of resource-bounded observation selection has been *myopic*. At any point in time, observations are planned as to satisfy the informativeness constraints for this time step, without considering the cost of making observations in the future. In this paper, we present an efficient *nonmyopic* algorithm for observation selection in spatio-temporal models. Within its planning horizon, our algorithm optimizes a collection of paths, one for each time step, which minimizes the long-term observation cost. More specifically, our contributions are:

- An efficient nonmyopic observation planning algorithm with strong theoretical performance guarantees.
- A general technique, which can use any myopic planning algorithm and convert it into a nonmyopic algorithm.
- Empirical analyses of the effectiveness of our algorithm on several real world data sets.

The NSTIP Problem

Our goal is to monitor a spatio-temporal phenomenon at a finite set of locations $\mathcal{V}$ and timesteps $\mathcal{T} = \{1, 2, \ldots, T\}$. With each location i and time t, we associate a random variable $\mathcal{X}_{i,t}$, and use $\mathcal{V}_t$ and $\mathcal{X}_{\mathcal{V}_t}$ to refer to all locations and their variables at time t, respectively. Since it is costly to observe all locations at every point in time, our approach will select a set of locations to visit at each time step, and use a statistical model to predict the missing values. To make this prediction, we assume we have a joint distribution $P(\mathcal{X})$ over all variables. This joint distribution encodes the dependencies along both spatial and temporal dimensions. In this paper, we use a class of nonparametric probabilistic models called Gaussian Processes (GPs, *c.f.,* Rasmussen & Williams, 2006), which have found successful use in modeling spatial phenomena such as temperature fields. Our approach is however not limited to this class of models.

In order to select which observations to make, we need to quantify the expected "informativeness" of these observations with respect to the missing ones. We quantify the informativeness of any set of observations $\mathcal{A} = \mathcal{A}_1 \cup \cdots \cup \mathcal{A}_t$ by some function $f(\mathcal{A})$, where $\mathcal{A}_t$ refers to the observations

made at time t. In the literature, there are different objective functions f which have been proposed to quantify informativeness, such as entropy (Shewry & Wynn 1987), mutual information (Caselton & Zidek 1984), or the reduction of predictive variance (Das & Kempe). In the case where no probabilistic model is available, one can also associate a sensing region observed by each sensor, and aim to maximize the total area covered (Bai *et al.* 2006). All these objective functions have two properties in common.[1] They are monotonic (i.e., $f(\mathcal{A}) \leq f(\mathcal{B})$ whenever $\mathcal{A} \subseteq \mathcal{B}$) and satisfy the following diminishing returns property: Adding a sensor to a small deployment helps more than adding it to a large deployment. More formally, $\forall \mathcal{A} \subseteq \mathcal{B} \subseteq \mathcal{V}$ and $s \in \mathcal{V} \setminus \mathcal{B}$, $f(\mathcal{A} \cup \{s\}) - f(\mathcal{A}) \geq f(\mathcal{B} \cup \{s\}) - f(\mathcal{B})$. A set function f satisfying this property is called *submodular* (*c.f.,* Nemhauser, Wolsey, & Fisher, 1978). Hence, in this paper we focus on the case of optimizing a monotonic submodular set function f.

In our spatio-temporal setting, we are solving the *filtering problem*: at each time step t, we would like to predict the values of all unobserved variables in the current time step based on information collected up to this point in time, $\mathcal{A}_{1:t} = \mathcal{A}_1 \cup \cdots \cup \mathcal{A}_t$. In particular, given a measure of informativeness for each time step, f_t, we would like to ensure that the information collected up to that time step passes some user-specified threshold k_t, i.e., $f_t(\mathcal{A}_{1:t}) \geq k_t$. For example, a user may require that the average standard deviation of estimates for each temperature reading at each time step is smaller than 1^oC.

For real-world applications like routing wireless packets or planning robot trajectories, chosen observations at each time t must be connected into a path $\mathcal{P}_t = (v_0 = s, v_1, \ldots, v_m = t)$, of nodes $v_i \in \mathcal{V}_t$, with start and end nodes s and t. For all time steps, we assume we are given a nonnegative, possibly asymmetric distance function $d_t : \mathcal{V}_t \times \mathcal{V}_t \rightarrow \mathbb{R}^+$. In the wireless sensor network example, d_t might reflect the expected transmission cost between any pair of locations at time t. The cost $C(\mathcal{P})$ of a path $\mathcal{P}$ can then be measured with respect to this distance function d_t.

We can now define our *nonmyopic spatiotemporal informative path planning problem (NSTIP)*. Given a collection of submodular functions $f_t : 2^{\mathcal{V}_1 \cup \cdots \cup \mathcal{V}_t} \rightarrow \mathbb{R}^+$ defined on the timesteps up to t, cost functions d_t, and a set of accuracy constraints k_t (e.g., on the desired variance reduction) we desire a collection of paths $\mathcal{P}_t$, one for each time step, with $\mathcal{P}^* = \operatorname{argmin}_{\mathcal{P}} \sum_{t=1}^{T} C(\mathcal{P}_t)$ subject to $f_t(\mathcal{P}_{1:t}) \geq k_t \ \forall t$. Hereby, $\mathcal{P}_{1:t} = \mathcal{P}_1 \cup \cdots \cup \mathcal{P}_t$, where $\mathcal{P}_{t'} \subseteq \mathcal{V}_{t'}$ is the path selected at timestep t'. (A path is properly a sequence, not a set, so the above is a minor abuse of notation.) We will assume in discussion that the start and end nodes for each path are a single specified base-station node, but our algorithms extend easily to settings where we do not need to return to a base station at each time step.

[1]Variance reduction is submodular under certain conditions (Das & Kempe). Mutual information is only approximately monotonic (Guestrin, Krause, & Singh 2005)

Nonmyopic Planning Algorithm

A naive approach for continuous querying is to treat each timestep as an independent single-step query, and optimize it independently. We will refer to this approach as *myopic*, as it does not look into the future. We are aiming for a *nonmyopic* approach, that performs optimization by adjusting the rewards of observations to account for the effect that these can have for later timesteps.

In our approach to the NSTIP problem, we use the following strategy. We first convert the problem of optimizing multiple paths, one for each timestep, into a problem of optimizing a single path on a new graph, the *nonmyopic planning graph* (NPG). We then show how one can use existing algorithms for solving a related problem, the *submodular orienteering problem* (SOP) (Chekuri & Pal 2005) as a subroutine to solve our nonmyopic planning problem. This procedure will retain the approximation guarantees which existing algorithms provide for the SOP for the myopic problem, while only introducing a small loss in the approximation guarantee due to the nonmyopic nature of our problem.

The Nonmyopic Planning Graph

Consider a candidate solution for the NSTIP problem consisting of a series of cyclic paths, one for each time step, each starting and ending at the base-station node. Imagine these plans arranged on a timeline and connected through their base-station nodes as in Fig. 1a.

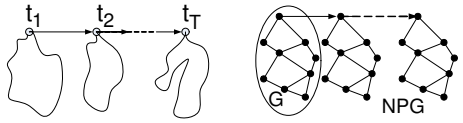


Figure 1: (a) Ex. NSTIP path. (b) Nonmyopic planning graph.

Fig. 1a represents the query plan across time. The paths connecting the basestation nodes are directed to demonstrate the fact that the plan for the first timestep is executed first, then it is followed by the plan for the second timestep, and so on. This overall solution, augmented by the edges connecting the basestation nodes at subsequent timesteps can now be considered a single path through a *different* graph. This *nonmyopic planning graph* (NPG) on *all* nodes $\mathcal{V}$ is constructed by combining the graph of finite-distance (via d_t) pairs of nodes from $\mathcal{V}_t$ for each timestep, adding zero-cost directed edges connecting them through the basestation nodes. This transformation is illustrated in Fig. 1b.

Our goal is to recover a solution to the NSTIP problem by optimizing a path in the NPG. Note that any path $\mathcal{P}$ through the NPG, starting at the basestation at time 1, and ending there at time T, uniquely corresponds to a collection of paths $\mathcal{P}_t$, one for each timestep. Moreover, the cost of $\mathcal{P}$ is exactly the sum of the costs of all $\mathcal{P}_t$. We now need to define an objective function f and a constraint k on the NPG, such that a solution $\mathcal{P}$ on NPG is feasible (i.e., $f(\mathcal{P}) \geq k$) iff the corresponding collection of paths is feasible (i.e., $f_t(\mathcal{P}_{1:t}) \geq k_t$). In order to achieve this, we set $k = \sum_t k_t$, and, for each timestep t, redefine a

function $f'_t(\mathcal{P}_{1:t}) = \min(f_t(\mathcal{P}_{1:t}), k_t)$. Hence, timestep t is satisfied iff $f'_t(\mathcal{P}_{1:t}) = k_t$. It can easily be verified that f'_t is still submodular and nondecreasing. Now define $f(\mathcal{P}) = \sum_t f'_t(\mathcal{P}_{1:t})$. f is nondecreasing and submodular, and $\mathcal{P}$ is a feasible solution iff $f(\mathcal{P}) = k$. Moreover, the set of optimal solutions coincides, having identical path costs.

Satisfying per-timestep constraints

Even after the transformation described in the previous section, the problem of finding a minimum cost feasible path on the NPG is still NP-hard.[2] Nonetheless, since it is expensive to collect observations, we would like to have an algorithm with theoretical guarantees to solve our problem. Unfortunately, we are unaware of the existence of any algorithm providing nontrivial guarantees for this problem. On the other hand, there are recent results by (Chekuri & Pal 2005) on a basically dual problem – the *submodular orienteering problem* (SOP). Instead of minimizing cost for a fixed information threshhold, SOP seeks a maximally-informative path $\mathcal{P}^*$ given a fixed budget B on path length. The cited paper presents a *recursive greedy* algorithm, which we call CP, which is guaranteed to return a path $\hat{\mathcal{P}}$ with cost at most B, such that $f(\hat{\mathcal{P}}) \geq \frac{1}{\alpha} f(\mathcal{P}^*)$, where $\alpha = \log |\mathcal{P}^*|$. Hence CP will return a solution where the reward is at most logarithmically worse in the length (number of nodes visited) of the optimal path. While the CP algorithm has the currently best known theoretical properties for SOP, our approach to NSTIP will accommodate *any* algorithm for SOP as a "black box". The approximation guarantee α will directly enter the approximation guarantees of our algorithm.

A naive approach to solve the NSTIP problem would be to call an SOP solver with increasing budgets, until we satisfy the reward constraints. Unfortunately, the algorithm for SOP only gives us an approximation guarantee for the reward of a particular budget; there is no guarantee offered on how much *more* budget is necessary to reach the desired reward.

Instead, we will stop our search as soon as we satisfy some *portion* of the total achievable reward. To save time, we increase the budget values by calling the algorithm for increasing powers of 2. In our setting, suppose that an optimal algorithm would fullfil the desired constraint ($f(P^*) = k$) with a path P^* with cost bounded by $2^j < C(P^*) \leq 2^{j+1}$, for some integer j. We then know that the approximate SOP algorithm, given a budget of budget $B = 2^{j+1}$ we will get a solution $\widehat{\mathcal{P}}_1$ with the guarantee $f(\widehat{\mathcal{P}}_1) \geq \frac{k}{\alpha}$.

However, since we covered only an α-fraction of the constraint k, we need to also cover the remaining difference. The key idea here is to look at the *residual reward*. For a set of "disqualified" nodes $\mathcal{A}$ (used in some earlier iteration), we define a new submodular function, $f_{\mathcal{A}}(\mathcal{B}) = f(\mathcal{A} \cup \mathcal{B}) - f(\mathcal{A})$. This new function $f_{\mathcal{A}}$ is also monotonic. Our goal is to cover the missing portion of the constraint by optimizing this residual function. That is, if we have found a path $\widehat{\mathcal{P}}_1$ in the first iteration, we will fulfill our constraint if we find a path $\mathcal{P}$ such that $f_{\widehat{\mathcal{P}}_1}(\mathcal{P}) = k - f(\widehat{\mathcal{P}}_1)$.

[2] In fact, a simple reduction from set cover shows that it is unlikely that any algorithm can achieve even a constant factor approximation to the optimal solution (Feige 1998).

Since f is a monotonic function, we know that $f_{\widehat{\mathcal{P}}_1}(\mathcal{P}^*) = k - f(\widehat{\mathcal{P}}_1)$. Hence, there must exist a path (e.g., $\mathcal{P}^*$) of budget at most B which covers the remaining reward $k - f(\widehat{\mathcal{P}}_1) \leq k - \frac{k}{\alpha}$ when optimizing $f_{\widehat{\mathcal{P}}_1}$, i.e., when planning *conditionally* on the nodes we already observed.

By the above argument, we can again find an approximate solution $\widehat{\mathcal{P}}_2$ which covers an α fraction of the remaining difference. Hence, after m iterations, the remaining difference is $k - f(\widehat{\mathcal{P}}_1 \cup \cdots \cup \widehat{\mathcal{P}}_m) \leq (1 - \frac{1}{\alpha})^m k$, which shrinks exponentially fast in m. This process is given in more detail in Algorithm 1. Hereby, $SOP_{\mathcal{A}}$ for a set of nodes $\mathcal{A}$ denotes a call to the submodular orienteering blackbox, using the residual reward $f_{\mathcal{A}}$.

Algorithm 1 $Cover(k)$

```
Budget = 0; k_cov = 0; P = ∅
while k_cov ≤ k(1 − ε) do
  for j = 1 ... j_max do
    B = 2^j; P̂' ← SOP_P(B)
    if f(P̂') ≥ (k − k_cov)/α then
      Budget = Budget + B; k_cov = k_cov + f_P(P̂)
      P = P ∪ P̂; break
```

Since the reward function f is real valued, we need to specify a threshold ϵ, such that we are satisfied with a solution which covers a $(1 - \epsilon)$ fraction of the desired accuracy constraint k. Theorem 1 bounds the number of iterations required in terms of the accuracy ϵ.

Theorem 1 *Algorithm 1 returns a solution of budget* $B \leq 2\frac{\log \epsilon}{\log(1-\frac{1}{\alpha})} B_{OPT}$ *which violates the constraint by at most* ϵk, *in time* $\mathcal{O}\left(\frac{\log \epsilon}{\log(1-\frac{1}{\alpha})} Q(nT, 2B_{OPT})\right)$.

Hereby, $Q(n, B)$ is the running time of the submodular orienteering black box executed on a graph with n nodes and budget B. For the CP algorithm, $Q(n, B) = \mathcal{O}((nB)^{\log(n)})$.

Efficient Nonmyopic Planning using Black Box

The approach presented in the previous section has running time proportional to $Q(nT, B)$, i.e., the running time of the submodular orienteering blackbox executed on a graph with nT nodes, and takes advantage of the algorithm's guarantee. Currently the best known guarantee is the one by Chekuri & Pal. Unfortunately, the guarantee of their approach comes at a price – albeit subexponential, the algorithm is superpolynomial: For our setting, their running time is bounded by $\mathcal{O}((nTB)^{\log(nT)})$. Using a spatial decomposition approach and branch and bound techniques, this running time can empirically be significantly decreased (Singh *et al.* 2007). However, the Nonmyopic Planning Graph gets very big very quickly, even for small horizons T, quickly rendering the approach infeasible. Instead, we can exploit the special structure of the Nonmyopic Planning Graph (NPG). This structure allows us to use dynamic programming (DP) to allocate the budget among the different timesteps, and then optimize paths for each timestep individually.

Our DP fills a $B \times T$ table $J(\cdot,\cdot)$. The entry at index (b,t) represents the best reward achieved up to and including timestep t for budget up to b. The DP function is then

$$J(b,t) = \max_{b' \leq b} \{J(b-b', t-1) + R(b', t|\widehat{\mathcal{P}}_{b-b',1:t-1})\} \quad (1)$$

Hereby, $R(b', t|\widehat{\mathcal{P}}_{b-b',t-1})$ denotes the reward obtained after running the SOP blackbox on the graph for the t-th timestep, optimizing the residual reward $f_{\widehat{\mathcal{P}}_{b-b',1:t-1}}$, conditional on the paths $\widehat{\mathcal{P}}_{b-b',1:t-1}$ selected in timesteps 1 through $t-1$, using a total budget of $b-b'$. (Path $\widehat{\mathcal{P}}_{b-b',1:t-1}$ is stored in cell $(b-b', t-1)$.) In addition to storing maximum reward in Eq. 1 into the cell (b,t) of the DP, we also remember the path that achieved this maximum. In particular, if the maximum was achieved by a budget level b'^*, we store the path $\widehat{\mathcal{P}}_{b-b'^*,1:t-1} \cup \widehat{\mathcal{P}}_{b'^*,t}$ in the same cell, where $\widehat{\mathcal{P}}_{b'^*,t}$ is the path returned by the SOP blackbox for budget b'^*. Since we store the selected path in each DP cell, we directly read the final solution from the DP table for time T.

Thus, our DP algorithm uses the SOP blackbox (e.g., the CP algorithm (Chekuri & Pal 2005)) to obtain a solution at each time step given each budget level b', conditioned on the paths selected in previous time steps using a total budget of $b-b'$. Note that this formulation of the DP is *greedy*: In order to fill the entries at time t, only observations made up to time t are taken into account. Since any observation can have effects on all future timesteps, this greedy choice disregards the observation choices we will make in future timesteps. We can compare two options: using the DP, with CP as blackbox (CP/DP), or apply CP to the NPG directly (CP/NPG). Perhaps surprisingly, the combination CP/DP is guaranteed to recover a solution which is at least as good as the solution obtained by CP/NPG, at orders of magnitude less running time. The reson for that is that the NPG graph has a special structure (is a chain of smaller graphs), forcing the CP/NPG solution to be separated into smaller subtours of specific budgets, allowing for a DP approach to work on the subgraphs individually.

Matrix element (b,t) can be computed after b calls to the SOP function which cost $\mathcal{O}(Q(n,B))$ each, for all choices of b' which could be up to B, so filling the entire matrix would require a runtime of $\mathcal{O}(B^2TQ(n,B))$.

Theorem 2 *In time $\mathcal{O}(\frac{\log \epsilon}{\log(1-\frac{1}{\alpha})} B^2TQ(n,B))$, the dynamic programming algorithm returns a solution of budget $B \leq 2\frac{\log \epsilon}{\log(1-\frac{1}{\alpha})} B_{OPT}$, that violates the constraint by at most ϵk.*

Note that when using CP/DP algorithm to optimize the paths for every timestep, for a fixed budget, the runtime of the DP is linear in T, whereas the runtime of CP/NPG grows superpolynomially with T. For example, in a network of 50 nodes, when called for a budget of 100, and an increase to the number of timesteps from 5 to 10, CP/NPG leads to a 10^7 factor of increase in the run time, compared to only a factor of 2 for CP/DP.

Adaptive Discretization

In order to compute an entry $J(b,t)$ of the DP table, we must perform an expensive call to the SOP blackbox for each value of $b' \leq b$. This computational cost quickly becomes prohibitive with increasing time-horizon T. To improve on the running time, we perform an *adaptive discretization*. Although the design choices are heuristic in nature, we found them to perform well empirically; a small number of budget levels (roughly on the order of the number of timesteps T) sufficed to achieve good solutions.

During a run of the algorithm, we dynamically decide which entries $J(b,t)$ to compute, as shown in Alg. 2. For each timestep t in order, we maintain a sorted list of budget levels $\tilde{\mathcal{B}}_t$. Initially, this list only contains only 0 and B. We also maintain the rewards of all budget levels. Now, we iteratively add a new budget level between levels b_i and b_{i+1}, such that $(rew(i+1) - rew(i)) \cdot (b_{i+1} - b_i)$ is maximized. The reason for this choice is that we simultaneously want to add discretization where the reward difference is large, but also where the gap between budgets is large.

In order to compute the reward for a given budget b at time t (function compRew of Alg. 2) we iterate through all budget levels $b_i \in \tilde{\mathcal{B}}_{t-1}$ from the previous timestep. For each b_i, we run the single-timestep optimization routine (SOP) with a budget level of $b - b_i$, conditional on the path associated with the previous cell. The maximum reward achieved becomes the reward of budget b at time t.

Algorithm 2 $DynDP(k,B,T)$

for $t = 1$ to T **do**
  $\tilde{\mathcal{B}}_t = \{b_0 = 0, b_1 = B\}\}$
  $rew[b_0]$=compRew(b_0); $rew[b_1]$ = compRew(b_1)
  for $j = 2$ to MaxNumLevels **do**
    $i = \arg\max_i (rew[b_{i+1}] - rew[b_i]) \cdot (b_{i+1} - b_i)$
    $b' = (b_{i+1} + b_i)/2$
    $rew[b']$ = compRew(b') using SOP blackbox
    insert b' into $\tilde{\mathcal{B}}_t$; store reward of b'

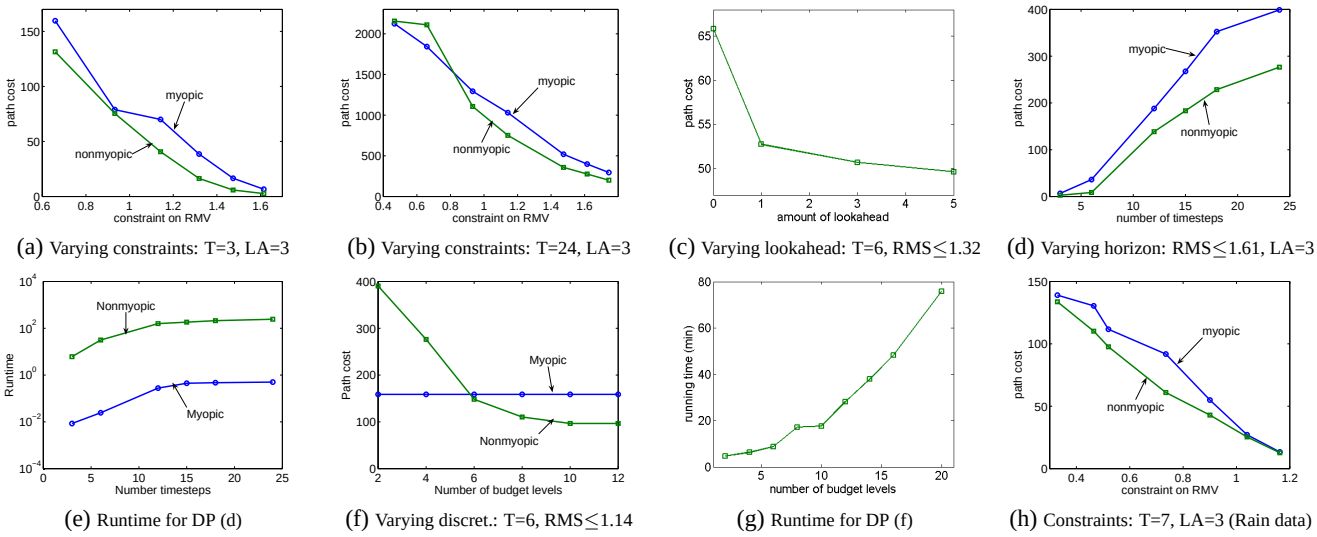
Evaluation

We empirically analyze our proposed algorithm, comparing the myopic and nonmyopic approaches for various settings of our parameters. We also demonstrate how these parameters affect the running time and the quality of the results.

Our experiments are based on two real data sets. Our first experiments use data gathered from a deployment of a 46 node sensor network at the Intel Berkeley Lab used by Deshpande *et al.*. Temperature and network connectivity measurements were gathered at one hour intervals, for a period of seven days. Five days of measurements were used for training and learning the transition model, and two were used for testing. We further verify our results with experiments on a different data set containing daily precipitation data from the Pacific Northwest of the United States, gathered from 37 stations over a period of 50 years (Widmann & Bretherton 1999). For both data sets, we learned Kalman Filter models, capturing the spatiotemporal correlations.

Implementation Details

In our experiments we wish to compare the myopic to the non-myopic approach. The myopic approach calls Alg. 1



(a) Varying constraints: T=3, LA=3 (b) Varying constraints: T=24, LA=3 (c) Varying lookahead: T=6, RMS$\leq$1.32 (d) Varying horizon: RMS$\leq$1.61, LA=3

(e) Runtime for DP (d) (f) Varying discret.: T=6, RMS$\leq$1.14 (g) Runtime for DP (f) (h) Constraints: T=7, LA=3 (Rain data)

Figure 2: (a-e) Myopic vs nonmyopic in terms of query plan cost and runtime. (f-g) Effects of adaptive discretization. (h) Rain data set.

once for every timestep, the objective function only considers the current timestep, and the model is updated based on the chosen observations, before proceeding to the next timestep. The nonmyopic algorithm uses the DP approach with adaptive discretization to produce an observation plan for multiple timesteps. The objective function considers the reward that an observation set has from the current timestep through a fixed number of timesteps into the future: $f'_t(\mathcal{A}_{1:t}) = \sum_{t'=t}^{\min(t+LA,T)} f_{t'}(\mathcal{A}_{1:t})$. Here, T is the planning horizon, LA is the lookahead.

Both algorithms use a fast heuristic by Chao, Golden, & Wasil as blackbox for the SOP problem, which Singh *et al.* experimentally showed to often provide relatively good results when compared to their efficient version of the recursive greedy algorithm of Chekuri & Pal.

In many practical applications, one wants to control the RMS error. Hence, in our implementation we chose the mean total variance reduction as our objective function: $f_t(\mathcal{A}_{1:t}) = \frac{1}{n}\sum_i (Var(\mathcal{X}_{i,t}) - Var(\mathcal{X}_{i,t} \mid \mathcal{X}_{\mathcal{A}_{1:t}}))$. This function is monotonically nondecreasing, but not always guaranteed to be submodular. There is both empirical and theoretical evidence however that in many practical problems, it indeed is submodular (Das & Kempe). Correspondingly, for our information constraints for every timestep, we chose to use the Root Mean Variance (RMV), since this is the criterion used in the myopic approach of Deshpande *et al.*, requiring $\sqrt{\frac{1}{n}\sum_i Var(\mathcal{X}_{i,t} \mid \mathcal{X}_{\mathcal{A}_{1:t}})} \leq k$, where k is the maximum RMV allowed each timestep.

Experiments

In our first set of experiments, we test how the accuracy constraint k affects the path cost achieved by both algorithms. Figures 2a and 2b display how the query path costs of the two algorithms change as we vary the constraints on the RMV. In both graphs, we used a lookahead of 3 steps in the objective function. The first graph presents plans for a horizon of 3 hours (or 3 timesteps), whereas the second uses a horizon of 24 hours. As expected, the path costs decrease as we loosen the constraints on the RMV. We observe that, apart from one data point, our nonmyopic algorithm always dominates the myopic one, providing better solutions. This outlier is due to the adaptive discretization procedure. Especially for rather loose constraints, we observe that the nonmyopic algorithm drastically outperforms the myopic algorithm, often providing a reduction in cost of up to 30%. We further note that for very loose constraints, as well as for very tight constraints, the performance of both algorithms is very similar. In the "loose" case, very few observations close to the basestation suffice, a solution found by both algorithms. In the "tight" case, both algorithms need to choose a large number of observations to satisfy the constraints, so the nonmyopic benefit is decreased.

With the experiment of Fig. 2c we examine how the lookahead parameter of the objective function affects the quality of the result for the nonmyopic approach. The experiment is performed for a planning horizon of 6 timesteps, and for a RMV constraint of 1.3. We see that, as the lookahead increases, the results get better, which is evidence that nonmyopic planning is very important for continuous queries. The biggest improvement however is achieved from the myopic setting for the objective function (lookahead 0) to a lookahead of 1.

In the final graph of this first set of experiments (Fig. 2d) we observe how the overall path cost of the two algorithms varies with the number of planning timesteps of the query. The RMV constraint used for this experiment was 1.6 and the lookahead of the non-myopic algorithm was set to 3. We observe that our non-myopic approach is consistently better. Also observe that the biggest improvement happens between 0 and 6 timesteps, when the nonmyopic algorithm drastically outperforms the myopic algorithm. This gain can be explained by noting that the experiment starts around midnight. In the first few hours, the model is very accu-

rate in predicting the temperatures from very few observations. During daytime hours, the path cost quickly increases as more observations have to be made. In the late evening to night time, again, few observations suffice for accurate predictions, and the nonmyopic algorithm again outperforms the myopic approach. In Fig. 2e we see how the running time of the DP is affected by the number of timesteps we need to plan for. As expected from our analysis of the running times, they grow linearly in terms of the horizon.

In our second set of experiments, we examine how the adaptive discretization in the DP affects the nonmyopic algorithm. Figures 2f and 2g display the results of this experiment for a horizon of six timesteps and a constraint on the RMS of 1.14. The first figure displays how the cost of the query plan produced by the non-myopic algorithm changes with the number of discretization levels. For very coarse discretization (few budget levels), the non-myopic solution is worse than the myopic result, because the coarse discretization prunes too much of the DP's search space, and good solutions are lost. Increasing the number of budget levels to roughly the order of timesteps, the solutions improve and exceed the performance of the myopic algorithm. As expected, finer discretization is more desirable, but as Fig. 2g shows, the running time quickly increases. This plot also shows that the non-adaptive approach quickly becomes intractable, as the running time of the DP is roughly proportional to B^2, which is of order $(nT)^2$. Extrapolating the performance observed in adaptive discretization, the runtime of the non-adaptive algorithm would be on the order of 10 days, compared to 20 minutes for the adaptive DP using 10 levels.

Finally, Fig. 2h displays results from the precipitation data. The experiment displays the cost of a 7 day horizon query, with a lookahead for the DP of 3 days, over varying constraints. The results are very similar to the ones we got from our previous dataset, and our non-myopic algorithm is again shown to outperform the myopic one, especially for intermediate tightness of the RMV constraint.

Related Work

The problem of data acquisition in sensing applications has been studied in literature. Deshpande *et al.* present the BBQ system, which proposes a model-driven scheme to provide approximate answers to queries posed in a sensor network, satisfying some information guarantees. This work focuses on the myopic setting, at each step the exact solution is obtained by an exponential algorithm, and a greedy heuristic is also provided, but with no approximation guarantees.

Liu, Petrovic, & Zhao consider the problem of nonmyopic collaborative target tracking in sensor networks. They nonmyopically optimize the information obtained from a sequence of observations. In order to optimize this criterion, they perform a heuristic "min-hop" search without approximation guarantees. Empirically their results shed more evidence on the importance of nonmyopic sensor selection.

Our problem is also related to the *Traveling Salesman Problem with profits* (TSPP; Feillet, Dejax, & Gendreau, 2005). In TSPP, each node has a fixed reward and the goal is to find a path that maximizes the sum of the rewards, while minimizing the cost of visited nodes. The orienteering problem is a special case of TSPP, maximizing rewards, subject to constraints on the cost (Laporte & Martello 1990). There are several important differences to this body of work. The TSPP objective, the sum of rewards, is a *modular* function. When selecting informative observations however, closeby locations are correlated, and hence their information is subadditive (submodular), making the problem more complex. Furthermore, our approach is nonmyopic, planning multiple paths, satisfying constraints for each time step. Our algorithm is efficient with respect to the planning horizon T, and provides approximation guarantees.

In robotics, similar work was developed in the context of *simultaneous localization and mapping* (SLAM). Stachniss, Grisetti, & Burgard develop a greedy algorithm, without approximation guarantees, for selecting the next location to visit to maximize information gain about the map. Sim & Roy 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.

Conclusions

In this paper, we addressed the problem of nonmyopically optimizing observation tours in spatiotemporal models. First, we provided a general technique, reducing the nonmyopic planning problem with accuracy constraints to be met at each timestep to the submodular orienteering problem (SOP) on a graph. Here, our approach allows us to use any SOP algorithm to be used as a blackbox. The approximation guarantees of the SOP blackbox are used by our algorithm to provide strong theoretical bounds about the cost of the paths obtained. We also develop an algorithm based on dynamic programming techniques that can reduce the cost of the nonmyopic planning by orders of magnitude. In addition, our adaptive discretization technique allows us to trade off solution quality and computational cost, empirically providing good solutions in a short amount of time. We demonstrate the effectiveness of our approach on two real-world data sets. Our results indicate that nonmyopic planning can drastically reduce the observation cost.

References

Bai, X.; Kumar, S.; Yun, Z.; Xuan, D.; and Lai, T. H. 2006. Deploying wireless sensors to achieve both coverage and connectivity. In *MobiHoc*.

Caselton, W., and Zidek, J. 1984. Optimal monitoring network design. *Statistics and Probability Letters*.

Chao, I.-M.; Golden, B. L.; and Wasil, E. A. 1996. A fast and effective heuristic for the orienteering problem. *Eur J Op Res*.

Chekuri, C., and Pal, M. 2005. A recursive greedy algorithm for walks in directed graphs. In *FOCS*.

Das, A., and Kempe, D. Algorithms for subset selection in linear regression. In *under review for STOC 2007*.

Deshpande, A.; Guestrin, C.; Madden, S.; Hellerstein, J.; and Hong, W. 2004. Model-driven data acquisition in sensor networks. In *VLDB*.

Feige, U. 1998. A threshold of ln n for approximating set cover. *J. ACM* 45(4).

Feillet, D.; Dejax, P.; and Gendreau, M. 2005. Traveling salesman problems with profits. *Transportation Science* 39(2).

Guestrin, C.; Krause, A.; and Singh, A. P. 2005. Near-optimal sensor placements in gaussian processes. In *ICML*.

Laporte, G., and Martello, S. 1990. The selective travelling salesman problem. *Discrete Appl. Math.* 26(2-3).

Liu, J.; Petrovic, D.; and Zhao, F. 2003. Multi-step information-directed sensor querying in distributed sensor networks. In *ICASSP*.

Meliou, A.; Chu, D.; Guestrin, C.; Hellerstein, J.; and Hong, W. 2006. Data gathering tours in sensor networks. In *IPSN*.

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

Rasmussen, C. E., and Williams, C. K. 2006. *Gaussian Processes for Machine Learning*. Adaptive Computation and Machine Learning. The MIT Press.

Shewry, M., and Wynn, H. 1987. Maximum entropy sampling. *J Appl Statist* 14.

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

Singh, A.; Krause, A.; Guestrin, C.; Kaiser, W.; and Batalin, M. 2007. Efficient planning of informative paths for multiple robots. In *IJCAI*.

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

Widmann, M., and Bretherton, C. S. 1999. 50 km resolution daily precipitation for the pacific northwest. http://www.jisao.washington.edu/data_sets/widmann/.

Proofs

Proof:[of Theorem 1] At every iteration of the while loop we cover an α-portion of the yet uncovered constraint, and the process will terminate when we have covered $k(1-\epsilon)$. At every step we are guaranteed not to exceed the budget of the previous step. This is because the optimal solution will always exist in the set of possible solutions that the SOP algorithm can pick. This optimal solution will have a reward that covers the constraints. Thus, based on the guarantees of the SOP algorithm, we know that for this budget the algorithm will return some set A' for which $f(A') \geq \frac{f(A_{OPT})}{\alpha}$. So, in every step, in order to cover an α-portion of the uncovered space, we will never need a budget bigger than 2^{j+1}, when the optimal budget is 2^j. This means that the SOP algorithm will never need to be called for a budget bigger than $2B_{OPT}$ if B_{OPT} is the budget of the optimal solution.

Also, at every step we aim for covering an α-portion of the uncovered constraint, so in iteration i the uncovered constraint would be $(1-\frac{1}{\alpha})^i k$. Since the algorithm will terminate when it has covered $\geq k(1-\epsilon)$, so the uncovered space would be $\leq k\epsilon$ (and thus the constraint cannot be violated by more than $k\epsilon$), we get that

$$(1-\frac{1}{\alpha})^i \leq \epsilon \Rightarrow i \leq \frac{\log \epsilon}{\log(1-\frac{1}{\alpha})}$$

So, we have a bound on the number of times that the while loop will be executed, which bounds the number of times that the SOP algorithm needs be called with the maximum budget ($2B_{OPT}$), in order to cover the reward constraints. If $Q(n,B)$ is the running time of the SOP blackbox for a graph of n nodes and for budget B, we know that we will call the SOP blackbox at most $\frac{\log \epsilon}{\log(1-\frac{1}{\alpha})}$ times on a graph of nT nodes and for budget $2B_{OPT}$.

Thus the running time of Algorithm 1 will be $O\left(\frac{\log \epsilon}{\log(1-\frac{1}{\alpha})} Q(n,B)\right)$

This also gives a bound on the total budget of the solution, since at every step we will never use more than $2B_{OPT}$ budget. So, our algorithm will give a solution with a budget no worse than $2\frac{\log \epsilon}{\log(1-\frac{1}{\alpha})} B_{OPT}$.

■

Proof:[of Theorem 2] If $Q(n,B)$ represents the running time of the SOP algorithm, for filling in a cell of the DP matrix we need to make up to B such calls, one for each possible division of the budget assignments between the current and all the previous timesteps. Also we have a total of BT cells in the matrix, so the DP has a running time of $O(B^2TQ(n,B))$.

The CP/DP algorithm will need to do i iterations to cover the reward constraint, where i can be bounded in the same way as in the previous proof:

$$i \leq \frac{\log \epsilon}{\log(1-\frac{1}{\alpha})}$$

This bounds the number of times that the DP will be called. Therefore, the running time of CP/DP would be $O(\frac{\log \epsilon}{\log(1-\frac{1}{\alpha})} B^2TQ(n,B))$.

The algorithm terminates when the covered constraint is $\geq k(1-\epsilon)$, so the constraint cannot be violated by more than $k\epsilon$.

■