Nonmyopic Informative Path Planning in Spatio-Temporal Models

Alexandra Meliou UC Berkeley
Andreas Krause CMU
Carlos Guestrin CMU
Joseph M. Hellerstein UC Berkeley

Abstract

In many sensing applications we must continuously gather information 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. The users issue queries with certain expectations on the answer quality. Thus, we must optimize the tours 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 nonmyopic greedy algorithm, exploiting submodularity of the information 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, based on its properties, provides strong theoretical guarantees for the solution. We also provide an extensive empirical analysis demonstrating the benefits of nonmyopic planning in a real world sensing application.

Introduction

Many practical applications require the monitoring of various physical phenomena that change over time. Sensing devices 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. 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*. 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 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 $\mathcal{T} = \{1, \ldots, T\}$. 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 selects a set of locations to visit at each time step, and uses a statistical model to predict the missing values. To make this prediction, we assume a joint distribution $P(\mathcal{X})$ over all variables. This joint distribution encodes the dependencies along 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. 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 $\mathcal{A} = \cup_{1..t}\mathcal{A}_j$ by some function $f(\mathcal{A})$, where $\mathcal{A}_t$ refers to the observations made at time t. In the literature, different objective functions f 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 2007). In the case where no probabilistic model is available, one can also associate a sensing region observed

Copyright © 2007, American Association for Artificial Intelligence (www.aaai.org). All rights reserved.

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., $f(\mathcal{A}) \leq f(\mathcal{B})$ for $\mathcal{A} \subseteq \mathcal{B}$) and satisfy the following diminishing returns property: Adding a sensor to a small deployment helps more than adding it to a large one. 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). In this paper we focus on optimizing a monotonic submodular set function f.

In our setting, we are solving the *filtering problem*: at each time 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, $\mathcal{A}_{1:t} = \cup_{1..t}\mathcal{A}_j$. 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.

Chosen observations at each time t must be connected into a path $\mathcal{P}_t = (v_0, \ldots, v_m)$ of nodes $v_i \in \mathcal{V}_t$. For all time steps, we assume we are given a nonnegative, possibly asymmetric distance function $d_t : \mathcal{V}_t \times \mathcal{V}_t \to \mathbb{R}^+$. 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 $f_t : 2^{\mathcal{V}_1 \cup \cdots \cup \mathcal{V}_t} \to \mathbb{R}^+$ defined on the timesteps up to t, cost functions d_t, and a set of accuracy constraints k_t 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, $\mathcal{P}_{1:t} = \cup_{1..t}\mathcal{P}_j$, where $\mathcal{P}_{t'} \subseteq \mathcal{V}_{t'}$ is the path selected at timestep t'. 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 to settings where we do not need to return to a base station at each time step.

Nonmyopic Planning Algorithm

A naive, *myopic*, approach for continuous querying is to treat each timestep as an independent single-step query, and optimize it independently. 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 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 to 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 SOP algorithms provide for the myopic case, while only introducing a small loss in the approximation guarantee due to the nonmyopic nature of our problem.

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

The Nonmyopic Planning Graph

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

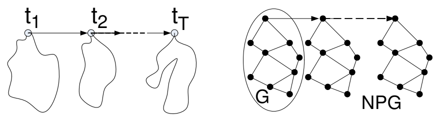


Figure 1: (a) Ex. NSTIP path. (b) Nonmyopic planning graph.

Fig. 1a represents the query plan across time. 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., $f_t(\mathcal{P}_{1:t}) \geq k_t$). 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 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 described transformation, the problem of finding a minimum cost feasible path on the NPG is still NP-hard (Feige 1998). Nonetheless, since it is expensive to collect observations, we need an algorithm with theoretical guarantees to solve our problem. Unfortunately, we are unaware 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 threshold, 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 would be to call an SOP solver with increasing budgets, until we satisfy the reward constraints. However, the SOP algorithm only gives an approximation guarantee for the reward of a particular budget, not 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 powers of 2. Suppose that an optimal algorithm would fulfill 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 $B = 2^{j+1}$ 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, $f_{\mathcal{A}}(\mathcal{B}) = f(\mathcal{A} \cup \mathcal{B}) - f(\mathcal{A})$,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 with a path $\mathcal{P}$ such that $f_{\widehat{\mathcal{P}}_1}(\mathcal{P}) = k - f(\widehat{\mathcal{P}}_1)$. 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.

Algorithm 1 $Cover(k)$

```
$Budget = 0$; $k_{cov} = 0$; $\mathcal{P} = \emptyset$
while $k_{cov} \leq k(1-\epsilon)$ do
  for $j = 1 \dots j_{max}$ do
    $B = 2^j$; $\widehat{\mathcal{P}}' \leftarrow SOP_{\mathcal{P}}(B)$
    if $f(\widehat{\mathcal{P}}') \geq \frac{k - k_{cov}}{\alpha}$ then
      $Budget = Budget + B$; $k_{cov} = k_{cov} + f_{\mathcal{P}}(\widehat{\mathcal{P}})$
      $\mathcal{P} = \mathcal{P} \cup \widehat{\mathcal{P}}$; break
    end if
  end for
end while
```

By the above argument, we can again find an approximate solution $\widehat{\mathcal{P}}_2$ which covers an α fraction of the remaining difference. 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 described 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 $f_{\mathcal{A}}$.

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 SOP 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

In the previous section we presented a transformation of a multistep problem to a single step equivalent, which allows for the use of existing algorithms with good theoretical bounds. Additionally we demonstrated how an approximation algorithm of a dual problem, the Submodular Orienteering Problem, can be adapted to address our setting, while preserving the algorithm's guarantees.

This approach 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 (2005). 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 (NPG) gets very big very quickly, even for small horizons T, quickly rendering the approach infeasible.

Nonmyopic Greedy Algorithm

In this section we will present an efficient greedy algorithm, which can be used more effectively in large structures like the Nonmyopic Planning Graph, and which also provides a theoretical bound better than that of Chekuri & Pal (2005) on the NPG. What makes the development of an efficient algorithm possible in this scenario is the flexibility of Algorithm 1, which was developed based on a blackbox understanding of the SOP Algorithm. Algorithm 1 simply uses a solution to the dual SOP problem, and transforms it to a solution of the NSTIP problem. What makes the algorithm computationally expensive is the use of the SOP algorithm on the NPG graph which is big in size. To develop an efficient alternative we will replace the blackbox that solves the SOP on NPG with another algorithm which is based on a greedy selection on every step, and resorts to the SOP algorithm on the smaller network graph.

In the same spirit as in our previous analysis, the reward that we gain from observing a set $\mathcal{A}$ is given by $f(\mathcal{A})$ which is a submodular monotone function. The marginal increase of f with respect to $\mathcal{A}$ and X is defined as $f'(\mathcal{A}; X) = f(\mathcal{A} \cup X) - f(\mathcal{A})$. All elements are chosen from a set $\mathcal{V}$, and for a positive budget B, our greedy blackbox will solve the budgeted maximization problem:

$$OPT = \operatorname*{argmax}_{\mathcal{A}\in\mathcal{V}:c(\mathcal{A})\leq B} f(\mathcal{A})$$

An element X in the greedy algorithm setting corresponds to a tour in the network graph at a particular timestep, and the approximately best tours are computed for each assignment of possible budgets per timestep. The SOP algorithm is used as an approximation oracle that returns a tour X with an α approximation factor relative to the optimal solution.

The details of our greedy algorithm are given in Algorithm 2.

Algorithm 2 $NonmyopicGreedy(k, Budget, T)$

```
$\mathcal{A}_1 = \emptyset$
$usedB = 0$
while $usedB \leq Budget$ do
  for $t = 1 \ldots T$ do
    for $b = 1 \ldots Budget - usedB$ do
      $\mathcal{M}(b,T) = SOP(b, \mathcal{G}_t, \mathcal{A}_1)$
    end for
  end for
  $X^* = \text{argmax}\{f'(\{\mathcal{A}_1; X_{b,t}\})/c(X_{b,t}) : X_{b,t} \in \mathcal{M}\}$
  $usedB = usedB + c(X^*)$
  $\mathcal{A}_1 = \mathcal{A}_1 \cup X^*$
end while
% Given $\mathcal{A}_1$ compute the best greedy choice for $Budget$
for $t = 1 \ldots T$ do
  $\mathcal{M}_2(Budget, t) = SOP(Budget, \mathcal{G}_t, \mathcal{A}_1)$
end for
$\mathcal{A}_2 = \text{argmax}\{f'(\{X_{b,t}\})/c(X_{b,t}) : X_{b,t} \in \mathcal{M}_2\}$
return $\text{argmax}_{\mathcal{A}\in\{\mathcal{A}_1,\mathcal{A}_2\}} f(\mathcal{A})$
```

The algorithm begins by filling in a $B \times T$ matrix, where element (b,t) contains the orienteering solution over the network graph that corresponds to timestep t and for budget b. The greedy rule adds to set $\mathcal{A}$ the element X^* such that

$$X^* = \max_{X\in\mathcal{W}\setminus\mathcal{G}_{i-1}} \frac{f'(\mathcal{G}_{i-1}; X)}{c(X_i)},$$

where $\mathcal{W}$ is the set of all possible choices for X and $\mathcal{G}_{i-1}$ the already chosen set. Since we can't evaluate the marginal increases $f'(\mathcal{G}_{i-1}; X)$ exactly, we only assume that we can evaluate $\hat{f}(\mathcal{G}_{i-1}; X)$, using the SOP algorithm as an approximation oracle. The main part of the algorithm is the computation of the greedy set of tours $\mathcal{A}_1$. However there are a few cases when $\mathcal{A}_1$ can be arbitrarily bad. These are covered by $\mathcal{A}_2$, which is the best tour when assigning all of the given budget to a single timestep, given set $\mathcal{A}_1$. $\mathcal{A}_2$ is guaranteed to be good in these few corner cases. Despite its greedy nature, our algorithm is still nonmyopic, as the reward functions consider the effect of node selections in future timesteps as well.

Algorithm 2 provides a solution to the SOP over multiple timesteps, which is equivalent to a solution on the NPG graph, and thus can replace the blackbox call to the SOP in algorithm 1. This is computationally an improvement, as the SOP procedure is now called on the smaller network graph as opposed to the NPG. In Algorithm 2 the initial population of the matrix requires $B \times T$ calls to the SOP, and the while loop will repeat this process at most B times. Thus, the running time of Algorithm 2 is $\mathcal{O}(B^2TQ(n,B))$, where $Q(n,B)$ is the running time of the SOP executed on a graph of n nodes and budget B. Specifically, if the CP algorithm is used for the SOP calls, the final running time will be $\mathcal{O}(B^2T(nB)^{\log n})$, as opposed to $\mathcal{O}((nTB)^{\log(nT)})$ that the call to the SOP on the NPG graph would cost.

Additionally to improving the running time, Algorithm 2, also provides better approximation bounds than the SOP on the NPG offers, which for the CP algorithm is $\frac{1}{\log(nT)}$. This statement is formally given by Theorem 2:

Theorem 2 *Algorithm 2 returns a solution path P for which*

$$f(P) \geq \frac{1}{2}\left(1 - \frac{1}{e^{\frac{1}{\alpha}}}\right) f(OPT) - \frac{1}{2}\beta\varepsilon$$

where α is the approximation factor of the SOP call.

More specifically, if the approximation factor of the SOP blackbox is $\log n$ as is the case with the CP algorithm, it is easy to show that $f(P) \geq \frac{1-e^{-1}}{2\log n} f(OPT)$.

Adaptive Discretization

To compute an entry $M(b,t)$ in Algorithm 2, we must perform an expensive call to the SOP blackbox for each value of $b \leq B$. This computational cost can quickly become prohibitive if we make the computation for all budget levels. To counter that we need a discretization scheme, and only make the expensive SOP call for some relatively small number of budget levels. Instead of choosing an arbitrary discretization approach, we propose an *adaptive discretization* which, although heuristic in nature, was found to perform well empirically; a small number of budget levels (roughly $O(T)$) sufficed to achieve good solutions.

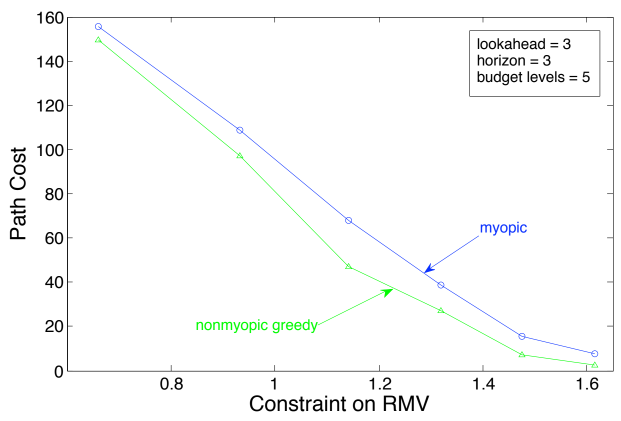
Algorithm 3 $Adaptive(k, B, T)$

```
for $t = 1$ to $T$ do
  $\tilde{\mathcal{B}}_t = \{b_0 = 1, b_1 = B\}\}$
  $rew[b_0]$=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'
  end for
end for
```

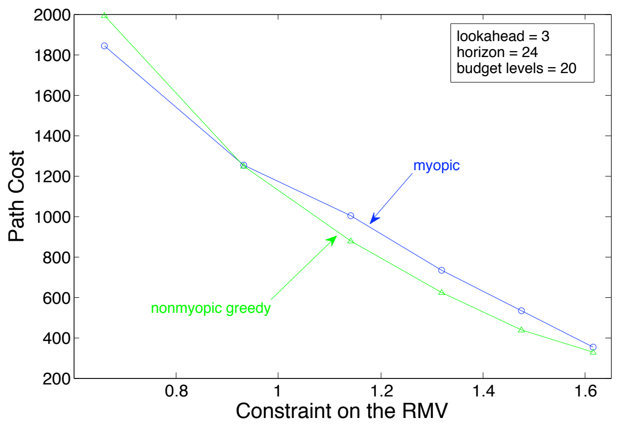
We dynamically decide which entries $M(b,t)$ to compute, as shown in Alg. 3. For each timestep t we maintain a sorted list of budget levels $\tilde{\mathcal{B}}_t$, initially containing only 1 and B, and the rewards of all budget levels. We iteratively add a new level between b_i and b_{i+1}, such that $(rew(i+1) - rew(i)) \cdot (b_{i+1} - b_i)$ is maximized. The reason is that we want to add discretization where the reward difference is large, but also where the gap between budgets is large.

Evaluation

We empirically analyze our proposed algorithm, comparing the myopic and nonmyopic greedy approaches for various

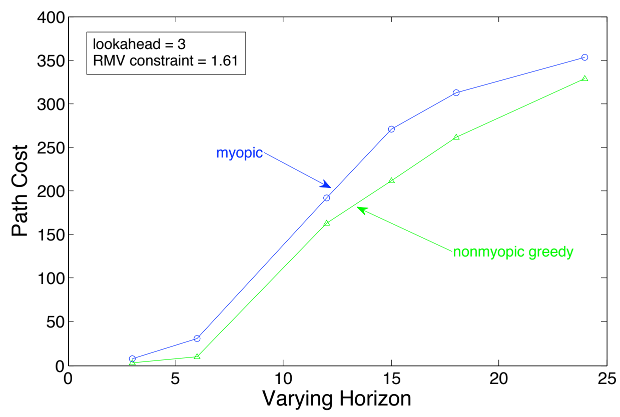


(a) Varying constraints: T=3, LA=3

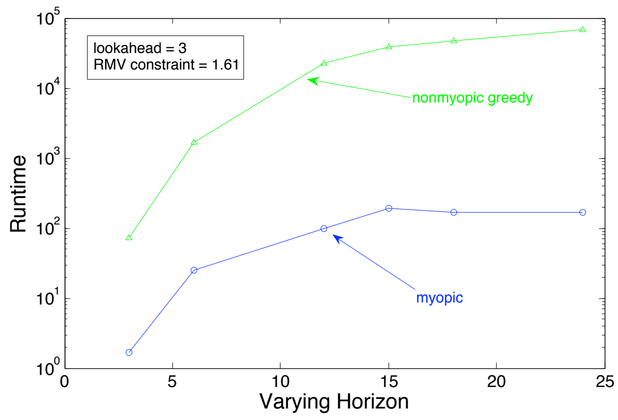


(b) Varying constraints: T=24, LA=3

Figure 2: Algorithm comparison for varying constraints

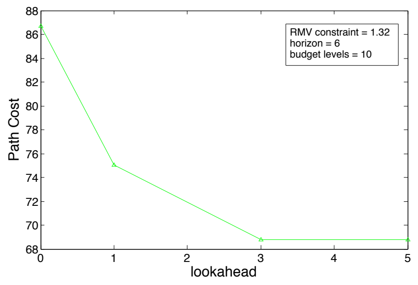


(a) Varying horizon: RMS≤1.61, LA=3

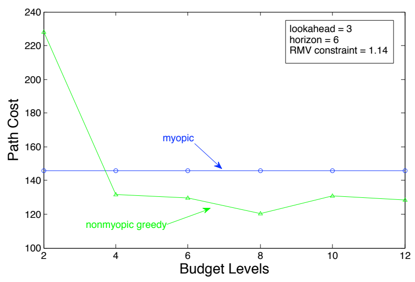


(b) Runtime comparison(d)

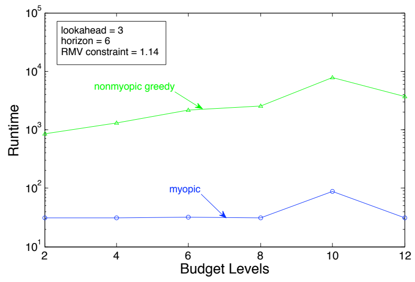
Figure 3: Algorithm comparison for varying horizon



(a) Varying lookahead: T=6, RMS≤1.32



(b) Varying discret.: T=6, RMS≤1.14



(c) Runtime for different budget levels (f)

Figure 4: Varying the parameters of the nonmyopic greedy algorithm

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 a real data set. Data was gathered from a deployment of a 46 node sensor network at the Intel Berkeley Lab used by Deshpande *et al.* (2004). Temperature and network connectivity measurements were gathered at one hour intervals, for a period of seven days. We learned Kalman filter models, capturing the satiotemporal correlations. Five days of measurements were used for training and learning the transition model, and two were used for testing.

Implementation Details

In our experiments we compare the myopic to the non-myopic greedy approach. The myopic approach calls Alg. 1 once for each 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 greedy approach with adaptive discretization to produce an observation plan for multiple timesteps. The objective function considers the reward that a set has from the current to a fixed number of timesteps into the future: $f'_t(\mathcal{A}_{1:t}) = \sum_{t'=t}^{\min(t+LA,T)} f_{t'}(\mathcal{A}_{1:t})$. T is the planning horizon, LA a lookahead parameter that allows us to interpolate between the myopic solution ($LA = 1$), to the completely nonmyopic approach ($LA = T$).

Both algorithms use a fast heuristic by Chao, Golden, & Wasil (1996) as blackbox for the SOP problem, which Singh *et al.* (2007) experimentally showed to often provide relatively good results when compared to their efficient version of the recursive greedy algorithm of Chekuri & Pal (2005).

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: $f_t(\mathcal{A}_{1:t}) = \frac{1}{n}\sum_i (Var(\mathcal{X}_{i,t}) - Var(\mathcal{X}_{i,t} \mid \mathcal{X}_{\mathcal{A}_{1:t}}))$, which is monotonically nondecreasing, but not always submodular. There is both empirical and theoretical evidence however that in many practical problems, it indeed is submodular (Das & Kempe 2007). So, for our per-timestep information constraints, we chose the Root Mean Variance (RMV), since this is the criterion used in the myopic approach of Deshpande *et al.* (2004), 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 both 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, whereas the second uses a horizon of 24 hours. The path costs decrease as we loosen the constraints on the RMV. 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. 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%. For very loose constraints, as well as for very tight constraints, the performance of both algorithms is very similar. In the "loose" case, few observations close to the basestation suffice, a solution found by both algorithms. In the "tight" case, both algorithms choose a large number of observations to satisfy the constraints, so the nonmyopic benefit is decreased.

In our second set of experiments (Fig. 3a) 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 greedy algorithm was set to 3. We observe that our non-myopic approach is consistently better. Relative to the number of timesteps, 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 accurate 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. 3b we see how the running time of the greedy algorithm 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 third set of experiments we examine how the various parameters of our greedy algorithm affect its performance. In Fig. 4a 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 is achieved from the myopic setting for the objective function (lookahead 0) to a lookahead of 1.

In he following two graphs, we examine how the adaptive discretization in the greedy algorithm affects its performance. Figures 4b and 4c 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 nonmyopic algorithm'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. 4c shows, the running time increases.

Related Work

The problem of data acquisition in sensing applications has been studied in literature. Deshpande *et al.* (2004) 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. At each step the exact solution is obtained by an exponential algorithm, under a myopic setting. A greedy heuristic is also provided, but with no approximation guarantees.

Liu, Petrovic, & Zhao (2003) consider the problem of nonmyopic collaborative target tracking in sensor networks. They nonmyopically optimize the information obtained from a sequence of observations. 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 is a *modular* function. When selecting informative observations however, closeby locations are correlated, and hence their information is sub-additive (submodular). 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 (2005) develop a greedy algorithm, without approximation guarantees, for selecting the next location to visit to maximize information gain about the map. Sim & 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.

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 single graph. Our approach allows for any SOP algorithm to be used as a blackbox. The approximation guarantees of the SOP blackbox are used to provide strong theoretical bounds about the cost of the paths obtained. We also develop a greedy algorithm, which still preserves the nonmyopic character of our approach, 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 a real-world data set. Our results indicate that nonmyopic planning can drastically reduce the observation cost.

Acknowledgments: This work was supported by NSF Grants CNS-0509383 and IIS-0205647, UC Micro grant 05-026 and gifts from Intel and Microsoft Corporations. Carlos Guestrin was supported in part by an IBM Faculty Fellowship, and an Alfred P. Sloan Fellowship. We also thank Amarjeet Singh for his implementation of the SOP heuristic and Stanislav Funiak for the model on the Intel Lab data.

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. 2007. Algorithms for subset selection in linear regression. In *under review for STOC*.

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*.

Meliou, A.; Krause, A.; Guestrin, C.; and Hellerstein, J. M. 2007. Nonmyopic informative path planning in spatio-temporal models. Technical Report UCB/EECS-2007-44, EECS, UC Berkeley.

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. 50km 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] Let us consider the computation of the set $\mathcal{A}_2$ in Algorithm 2. Name $\mathcal{V}$ all the possible choices of X, i.e. every element of the matrix M. Renumber $\mathcal{V} = \{X_1, \ldots, X_n\}$ and define $\mathcal{G}_0 = \emptyset$ and $\mathcal{G}_i = \{X_1, \ldots, X_i\}$ such that

$$\frac{f(\mathcal{G}_i) - f(\mathcal{G}_{i-1})}{c(X_i)} \geq \max_Y \frac{f(\mathcal{G}_{i-1} \cup Y) - f(\mathcal{G})}{\alpha c(Y)}$$

for some $\alpha \geq 1$, which is the approximation factor of the approximation oracle. The sequence $(G_j)_j$ corresponds to the sequence of assignments to $\mathcal{A}_2$, and is motivated by the simple greedy rule, adding, for a prior selection $\mathcal{G}_{i-1}$, the element X_i such that

$$X_i = \max_{X \in \mathcal{W} \setminus \mathcal{G}_{i-1}} \frac{\hat{f}(\mathcal{G}_{i-1}; X)}{c(X_i)}.$$

Let $l = \max\{i : c(\mathcal{G}_i) \leq B\}$ be the index corresponding to the iteration, where $\mathcal{A}_2$ is last augmented, hence $\mathcal{A}_2 = \mathcal{G}_l$. Let $L = c(OPT)$, $c_{min} = \min_X c(X)$ and $w = |OPT|$.

To prove Theorem 2, we need two lemmas:

Lemma 1 *For $i = 1, \ldots, l+1$, it holds that*

$$f(\mathcal{G}_i) - f(\mathcal{G}_{i-1}) \geq \frac{c(X_i)}{\alpha L}(f(OPT) - f(\mathcal{G}_{i-1})) - \varepsilon\left(1 + \frac{wc(X_i)}{\alpha L}\right)$$

Proof: Using monotonicity of f, we have

$$f(OPT) - f(\mathcal{G}_{i-1}) \leq f(OPT \cup \mathcal{G}_{i-1}) - f(\mathcal{G}_{i-1}) = f(OPT \setminus \mathcal{G}_{i-1} \cup \mathcal{G}_{i-1}) - f(\mathcal{G}_{i-1})$$

Assume $OPT \setminus \mathcal{G}_{i-1} = \{Y_1, \ldots, Y_m\}$, and let for $j = 1, \ldots, m$

$$Z_j = f(\mathcal{G}_{i-1} \cup \{Y_1, \ldots, Y_j\}) - f(\mathcal{G}_{i-1} \cup \{Y_1, \ldots, Y_{j-1}\}).$$

Then $f(OPT) - f(\mathcal{G}_{i-1}) \leq \sum_{j=1}^m Z_j$.

Now notice that

$$\frac{Z_j - \varepsilon}{\alpha c(Y_j)} \leq \frac{f(\mathcal{G}_{i-1} \cup Y_j) - f(\mathcal{G}_{i-1}) - \varepsilon}{\alpha c(Y_j)} \leq \frac{f(\mathcal{G}_i) - f(\mathcal{G}_{i-1}) + \varepsilon}{c(X_i)}$$

using submodularity in the first and the greedy rule in the second inequality. Since $\sum_{j=1}^m c(Y_j) \leq L$ it holds that

$$f(OPT) - f(\mathcal{G}_{i-1}) = \sum_{j=1}^m Z_j \leq \alpha L \frac{f(\mathcal{G}_i) - f(\mathcal{G}_{i-1}) + \varepsilon}{c(X_i)} + m\varepsilon$$

■

Lemma 2 *For $i = 1, \dots, l+1$ it holds that*

$$f(\mathcal{G}_i) \geq \left[1 - \prod_{k=1}^{i}\left(1 - \frac{c(X_k)}{\alpha L}\right)\right] f(OPT) - \left(\frac{\alpha L}{c(X_i)} + w\right)\varepsilon.$$

Proof: Let $i = 1$ for sake of induction. We need to prove that $f(\mathcal{G}_1) \geq \frac{c(X_1)}{\alpha L} f(OPT) - \left(\frac{\alpha L}{c(X_i)} + w\right)\varepsilon$. This follows from Lemma 1 and since

$$\frac{\alpha L}{c(X_i)} + w \geq 1 + \frac{wc(X_i)}{\alpha L}.$$

Now let $i > 1$. We have

$$\begin{aligned}
f(\mathcal{G}_i) &= f(\mathcal{G}_{i-1}) + [f(\mathcal{G}_i) - f(\mathcal{G}_{i-1})] \\
&\geq f(\mathcal{G}_{i-1}) + \frac{c(X_i)}{\alpha L}[f(OPT) - f(\mathcal{G}_{i-1})] - \varepsilon\left(1 + \frac{wc(X_i)}{\alpha L}\right) \\
&= \left(1 - \frac{c(X_i)}{\alpha L}\right) f(\mathcal{G}_{i-1}) + \frac{c(X_i)}{\alpha L} f(OPT) - \varepsilon\left(1 + \frac{wc(X_i)}{\alpha L}\right) \\
&\geq \left(1 - \frac{c(X_i)}{\alpha L}\right)\left[\left(1 - \prod_{k=1}^{i-1}\left(1 - \frac{c(X_k)}{\alpha L}\right)\right) f(OPT) - \left(\frac{\alpha L}{c(X_i)} + w\right)\varepsilon\right] + \frac{c(X_i)}{\alpha L} f(OPT) - \varepsilon\left(1 + \frac{wc(X_i)}{\alpha L}\right) \\
&= \left(1 - \prod_{k=1}^{i}\left(1 - \frac{c(X_k)}{\alpha L}\right)\right) f(OPT) - \varepsilon\left(1 + \frac{wc(X_i)}{\alpha L}\right) - \left(\frac{\alpha L}{c(X_i)} + w\right)\varepsilon\left(1 - \frac{c(X_i)}{\alpha L}\right) \\
&= \left(1 - \prod_{k=1}^{i}\left(1 - \frac{c(X_k)}{\alpha L}\right)\right) f(OPT) - \left(\frac{\alpha L}{c(X_i)} + w\right)\varepsilon
\end{aligned}$$

using Lemma 1 in the first and the induction hypothesis in the second inequality. ■

From now on let $\beta = \frac{\alpha L}{c_{min}} + w$.

Observe that for $a_1, \dots, a_n \in \mathbb{R}^+$ such that $\sum a_i = A$, the function $(1 - \prod_{i=1}^{n}(1 - \frac{a_i}{\alpha A}))$ achieves its minimum at $a_1 = \cdots = a_n = \frac{A}{n}$. In order to show this, let $G(a_1, \dots, a_{m-1}) = \log \frac{A - \sum_{j=1}^{m-1} a_j}{A} + \sum_i \log(\alpha - \frac{a_i}{A})$. It holds that

$$\frac{\partial G}{\partial a_i} = \frac{-1/A}{\alpha - a_i/A} + \frac{1/A}{\alpha - (A - \sum_{j=1}^{m-1} a_j)/A}.$$

This derivative is 0 iff $A - \sum_{j=1}^{m-1} a_j = a_i$ for all $1 \leq i < m$. Hence, for $1 \leq i \leq m$ it must hold that $a_i = A/m$. We hence have

$$\begin{aligned}
f(\mathcal{G}_{l+1}) &\geq \left[1 - \prod_{k=1}^{l+1}\left(1 - \frac{c(X_k)}{\alpha L}\right)\right] f(OPT) - \beta\varepsilon \\
&\geq \left[1 - \prod_{k=1}^{i}\left(1 - \frac{c(X_k)}{\alpha c(\mathcal{G}_{l+1})}\right)\right] f(OPT) - \beta\varepsilon \\
&\geq \left[1 - \left(1 - \frac{1}{\alpha(l+1)}\right)^{l+1}\right] f(OPT) - \beta\varepsilon \\
&\geq \left(1 - \frac{1}{e^{1/\alpha}}\right) f(OPT) - \beta\varepsilon
\end{aligned}$$

where the first inequality follows from Lemma 2 and the second inequality follows from the fact that $c(\mathcal{G}_{l+1}) > L$, since it violates the budget.

Name X_{l+1} the second candidate solution considered by the algorithm. From submodularity and monotonicity we get:

$$f(\mathcal{G}_l) + f(X_{l+1}) \geq f(\mathcal{G}_{l+1}) \geq \left(1 - \frac{1}{e^{1/\alpha}}\right) f(OPT) - \beta\varepsilon$$

thus

$$\max\{f(\mathcal{G}_l), f(X_{l+1})\} \geq \frac{1}{2}\left(1 - \frac{1}{e^{1/\alpha}}\right) f(OPT) - \beta\varepsilon$$

■

Specifically, for $\alpha = \log n$, the bound becomes as follows:

Since n represents the nodes in the network, n is an integer ≥ 2. Now, $\forall x \in [0,1]$ and $\forall \eta$, it is $e^{\eta x} \leq 1 + (e^{\eta} - 1)x$. Replacing x with $\frac{1}{\log n}$ and $\eta = -1$ we get:

$$e^{-\frac{1}{\log n}} \leq 1 + (e^{-1} - 1)x \Rightarrow 1 - \frac{1}{e^{\frac{1}{\log n}}} \geq (1 - e^{-1})\frac{1}{\log n}$$

Using this in the previous bound we get:

$$f(P) \geq \frac{1 - e^{-1}}{2 \log n} f(OPT)$$