Stochastic Steepest-Descent Optimization of Multiple-Objective Mobile Sensor Coverage

Chris Y. T. Ma, David K. Y. Yau, Nung Kwan Yip
Purdue University
West Lafayette, IN, USA

Nageswara S. V. Rao
Oak Ridge National Lab
TN, USA

Jiming Chen
Zhejiang University
Hangzhou, China

***Abstract*—We propose a steepest descent method to compute optimal control parameters for balancing between multiple performance objectives in stateless stochastic scheduling, wherein the scheduling decision is effected by a simple constant-time coin toss operation only. We apply our method to the scheduling of a mobile sensor's coverage time among a set of points of interest (PoIs). The coverage algorithm is guided by a Markov chain wherein the sensor at PoI i decides to go to the next PoI j with transition probability p_{ij}. We use steepest descent to compute the transition probabilities for optimal tradeoff between two performance goals concerning the distributions of per-PoI coverage times and exposure times, respectively. We also discuss how other important goals such as energy efficiency and entropy of the coverage schedule can be addressed. For computational efficiency, we show how to optimally adapt the step size in steepest descent to achieve fast convergence. However, we found that the structure of our problem is complex in that there may exist surprisingly many local optima in the solution space, causing basic steepest descent to get stuck easily at a local optimum. To solve the problem, we show how proper incorporation of noise in the search process can get us out of the local optima with high probability. We provide simulation results to verify the accuracy of our analysis, and show that our method can converge to the globally optimal control parameters under different assigned weights to the performance goals and different initial parameters.**

I. Introduction

It is known that stochastic scheduling techniques can achieve service differentiation by maintaining minimal scheduling state. For example, stochastic fair queuing (SFQ) [1] allows a router to allocate weighted service rates among flows without bookkeeping of past service received by each flow. In SFQ, to give p_i ($0 \leq p_i \leq 1$) fraction of service to flow i, we toss a coin and with probability p_i, serve the next packet from i. SFQ has a constant-time computational complexity. In contrast, the deterministic scheduling technique of weighted fair queueing (WFQ) [2] selects the next flow for service based on a sorting of per-flow finish numbers. Bookkeeping the finish numbers has a computational complexity of $O(n^2)$ for n flows. For CPU time sharing, lottery scheduling [3] is a probabilistic algorithm that allows processes to receive long-term CPU rates in proportion to their assigned numbers of lottery tickets. In contrast, stride scheduling [4], a deterministic implementation of the lottery scheduling concept, selects processes to run based on a global pass number and per-process pass numbers similar to WFQ scheduling variables.

Besides service rates, fairness is an important scheduling metric. Fairness determines how long a client, after receiving any amount of service, has to wait before selected again for service. For example, a flow that is given the first of every ten packet times of service has the same rate as another flow given the first two of every 20 packet times of service. However, the first flow achieves finer grained fairness compared with the second flow. Deterministic schedulers can be designed for different fairness goals independent of the service rates. For example, for the same set of rate guarantees to flows, WF^{2}Q [5] is designed to achieve finer grained fairness than WFQ. On the other hand, existing stochastic schedulers do not try to control independently the rate and fairness performance. For example, to increase the frequency at which flow i is selected for service in SFQ, one has to increase p_i, but doing so will also increase the service rate of i. In other words, there is a coupling between rate and fairness in SFQ.

In this paper, we propose a steepest descent method to compute control parameters for achieving an optimal tradeoff between multiple objectives in stateless stochastic scheduling. By stateless, we mean that the scheduling algorithm does not need any bookkeeping of prior service received by clients, but the scheduling decision is determined entirely by a constant-time coin toss operation. The notion of optimal balance between multiple performance objectives is user- or application-defined. For example, some applications may give a larger importance to rate than fairness, whereas other applications may require the opposite. Our ability to support a wide range of multi-objective performance tradeoffs and optimize the tradeoff is absent in existing stochastic scheduling algorithms.

We apply our method to the problem of scheduling a mobile sensor's coverage time among a number of disjoint *points of interest* (PoIs). In this case, the sensor moves between the PoIs for capturing interesting information, and the notions of rate and fairness have similarly important interpretations. Specifically, rate corresponds to the fraction of time that the sensor covers a PoI. The larger the fraction of time, the more information the sensor can collect about the PoI. Fairness, on the other hand, determines the average return time of the sensor to a PoI. The longer the return time, the longer the PoI can be exposed to significant incidents that have occurred but are left undetected. The delay in responding to an incident, say an accident that requires rescue operations, may increase. An important issue in our problem context is that the solution space is extremely complex due to a large number of control parameters affecting multiple performance goals. Particularly, there may exist a surprisingly large number of local optima, and noise in the optimization procedure is essential to avoid getting stuck at these local optima. We will show that our stochastic optimization method can optimally balance between the coverage time and exposure time metrics. A reduced weight on the exposure time metric will allow the sensor to move less, which has the effect of conserving motion energy. More generally, we will discuss how our approach can address explicitly energy efficiency and other performance objectives such as the entropy of the coverage schedule (increased entropy implies increased randomness, which may be important for the detection of smart adversaries, by making it harder for the adversaries to anticipate the sensor's location and avoid detection). To the best of our knowledge, there is no existing coverage algorithm – deterministic or stochastic, and constant-time or otherwise – that can provide such optimal tradeoffs.

The case for mobility in monitoring applications has been extensively discussed in the literature [6], [7], [8], [9]. In general, mobile nodes are useful in situations where the installation of extensive static sensing/communication devices is difficult. For example, chemical sensors can be deployed at specific locations in a water distribution system (WDS) to monitor the water quality for the growth of biological masses, the presence of contaminants, etc [10], [11], [12]. Long-range wireless communication under water is, however, infeasible due to high signal attenuation. To solve the problem, a mobile node can be used to move among the underwater sensors, gather data from the sensors, and carry the data to a sink for reporting. In this case, it has been recognized that the overall optimization problem has multiple objectives that may be antagonistic [12]. For example, focused data collection near the periphery of the WDS (specifically, immediate downstream of likely entry points of contaminants) may minimize the delay of detection, but focused data collection at the center of the WDS may maximize the detection probability.

A. Our contributions

Our contributions in this paper are as follows.

(i) We analyze a general stochastic algorithm to schedule a mobile sensor's coverage time among a set of disjoint PoIs denoted by S. The scheduling is controlled by a discrete-time Markov chain taking values in S, wherein the sensor at PoI i decides to move to PoI j in the next time step with transition probability p_{ij}. (If $i = j$, the sensor stays at the same PoI.) The performance of the Markov chain is measured by its time limiting measure of a cost function. The key feature of our setting is that the cost reflects conflicting criteria such as the distributions of Per-PoI coverage and exposure times. In addition, our Markov model accounts for the physical constraints associated with geographical placements of the PoIs. In particular, travel from one PoI to another must occur along a physically feasible route and is subject to speed constraints. The sensor may also go through other PoIs that are on the route, thereby affecting the coverage time of these intermediate PoIs.

(ii) We provide closed-form analytical formulas in terms of the Markov transition probabilities to express the cost function. To provide a specific model example, we consider the criteria of (1) the achieved distribution of coverage times among the PoIs, and (2) the achieved distribution of the per-PoI exposure times. Our formulation can be easily extended to include other physical factors such as energy efficiency, entropy, amount of information captured, etc.

(iii) We allow a target allocation of the sensor's coverage time among the PoIs to be specified. The scheduler then attempts to balance between two performance metrics: (1) deviation of the distribution of coverage times from the target allocation, and (2) the average return time of the sensor to the PoIs. Based on our analytical formulas, we develop a steepest descent optimization method to determine the Markov transition probabilities such that the cost function, expressed as a weighted sum of the dual performance metrics, is minimized. The optimization is comprehensive in that we search for an optimal solution in the space of *all* transition probabilities.

The trend of our optimal solutions agrees with the physical interpretation. In particular, we show that as we reduce the weight of the exposure time metric, the sensor can adapt by moving less between the PoIs, thereby conserving motion energy and achieving more closely the target per-PoI allocation of coverage times.

(iv) As a practical numerical computation issue, we discuss how to estimate optimal time steps in the steepest descent to achieve fast convergence to the optimal Markov transition probabilities.

(v) We present empirical evidence that there may exist many local optima in the solution space of our problem, due to our comprehensive search space of all the transition probabilities for multiple performance criteria. Hence, the basic steepest descent method may become trapped at a local optimum with high probability, i.e., from more than 60% of the random positions where the search starts. We show that by properly adding noise in the search process, we can get out of the local optima in practically all the test cases, arriving at the global optimum solution irrespective of where the search began.

All of the above assertions are verified by robust numerical and simulation results.

II. Related Work

On-line scheduling of our mobile sensor is controlled by a Markov chain taking values in a set of PoIs covered by the sensor. When the sensor is at a PoI, say i, it determines the next PoI j to visit with transition probability p_{ij}. Hence, the on-line scheduling is effected by a simple coin toss at each decision point. The approach is similar to existing scheduling algorithms such as stochastic fair queuing (SFQ) [1] and lottery scheduling [3]. However, in SFQ and lottery scheduling, determining the probabilities of the coin toss is simple and based on only the target service rates for the clients. In our work, we aim to optimize the transition probabilities to achieve a user-/application-defined notion of the best tradeoff between a number of possibly conflicting performance objectives. This optimization of the transition probabilities via stochastic steepest descent represents a fundamental departure from existing scheduling approaches.

Markov Chain Monte Carlo (MCMC) methods [13], [14] can be used to determine the Markov transition probabilities if we target *only* the distribution of the sensor's coverage time among the PoIs, although care is needed to account for the intermediate PoIs visited as the sensor moves from the current PoI i to the next PoI j. In this case, the solution that achieves a target coverage-time distribution is not unique, and can be optimized for maximum entropy. MCMC methods are not applicable when the tradeoffs between multiple performance objectives are considered.

The case for multiple performance objectives in scheduling problems has been well recognized. Rate, fairness, delay, and jitters are common performance metrics for a range of time-shared resources such as communication and computation bandwidth. Given one of the performance goals, it is known that different algorithms may be designed to achieve different performance for another goal [15]. In sensor networks, the importance of multiple performance objectives and the challenge in balancing between the possibly conflicting objectives have been similarly acknowledged [12]. For the placement of static sensors against contaminations in a water distribution system (WDS), for example, different optimization objectives will lead to different placement strategies. To minimize the delay of detecting contaminants in the WDS, placement of sensors at the periphery, where the contaminants will likely enter the WDS, is called for. To maximize the probability of detection, however, placement in the middle of the WDS may be the most effective.

However, while the existence of diverse performance objectives is well known, and individual algorithms may approach their simultaneous achievement differently, the design of a uniform approach to support a range of tradeoffs and optimize the tradeoff according to user/application requirements has been absent in the literature. Our work aims to provide such an approach in optimizing tradeoffs between important objectives in mobile sensor coverage, including the distributions of per-PoI coverage times and exposure times, energy efficiency, and entropy of the coverage schedule. To the best of our knowledge, there is no existing approach that can provide these optimal tradeoffs in sensor coverage problems.

III. Physical Set-Up

We consider a mobile sensor moving among a number of *points of interest* (PoIs) placed in a geographical area. The sensor has a sensing range of r. A PoI, say i, is said to be *covered* by the sensor if i is within distance r of the sensor, in which case the sensor can gather useful information about i (e.g., detect any interesting event happening at i). The PoIs are disjoint in that no two PoIs can be covered by the sensor at the same time. However, they are fully connected meaning that a path exists for the sensor to move between any two PoIs. As the sensor travels from PoI i to PoI j on a path, it may pass through (and hence cover) other intermediate PoIs that are situated geographically along the path. We allow the user to specify a target allocation, denoted by Φ, of the sensor's coverage time among the PoIs, where Φ_i is the target fraction of the coverage time that should be received by i. The target allocation may reflect the different importance of the PoIs.

Scheduling of our mobile sensor is controlled by a Markov chain, which describes the sensor's travel between the PoIs. We design the Markov chain to minimize a general cost function, which can account for factors such as the distributions of per-PoI coverage and exposure times, an entropy measure quantifying the randomness of the sensor schedule, information capture about dynamic events, energy efficiency, etc. A key feature of our framework is that it can incorporate possibly conflicting performance objectives while accounting for the dependencies between the PoIs covered due to their geographical placement.

For the sake of discussion, we will focus on a cost function, denoted as U, that considers the distributions of per-PoI coverage times and per-PoI exposure times. Other performance criteria can be included without much difficulty. Our approach expresses the cost function in terms of the transition probabilities of the Markov chain and uses steepest descent to find the optimal transition probabilities. The advantages of our method are its *simplicity* and the *comprehensiveness of the search space*.

A. Derivation of the cost function

We now describe details of the mathematical set-up. Suppose there are M PoIs, denoted by the set $S = \{1, 2, \ldots M\}$. Consider a *time-homogeneous* Markov chain $\{X_n\}_{n\geq 1}$ taking values in S. The sensor moves between the PoI locations (i.e., states in the Markov chain) as the Markov chain state transitions. The transition probabilities are denoted by $\{p_{ij}\}_{i,j\in S}$: $p_{ij} = P(X_{n+1} = j | X_n = i)$.

In order to relate the (discrete) transition time step n and the physical elapsed time, we use the following notation:

- T_{jk} equals the *sum* of the *travel time* from PoI j to PoI k and the *pause time* P_k at k. Note that $T_{jj} = P_j$.
- $T_{jk,i}$ equals the time the sensor, upon traveling from j to k, passes by PoI i. These quantities in essence reflect the side effects of coverage imposed by geographical constraints: A PoI, say i, can still be covered even if it is not the intended destination of a transition, because the sensor has to go through i in order to reach the destination. We use the following convention: $T_{jj,i} = 0$ for $j \neq i$ and $T_{jj,j} = P_j$ (the pause time at j).

With the above, the physical time elapsed after N state transitions is given by:

$$T(N) = \sum_{n=1}^{N} \sum_{j,k} T_{jk} \delta_j(X_n) \delta_k(X_{n+1}),$$

where $\delta_i(x) = 1$ if $x = i$ and 0 otherwise. Furthermore, the total coverage time of PoI i by the sensor is given by:

$$C_i(N) = \sum_{n=1}^{N} \sum_{j,k} T_{jk,i} \delta_j(X_n) \delta_k(X_{n+1}).$$

Next we define $\langle E_i(N) \rangle$—the average *exposure time*—to be the *arithmetic average* (taken in the first N transitions) of the lengths of the *continuous* time intervals during which the PoI i is *out of range* of the sensor. More precisely, let $E_{i,1}, E_{i,2}, \ldots E_{i,m}$ be the consecutive time intervals during which the sensor is away from i. Then

$$\langle E_i(N) \rangle = \frac{E_{i,1} + E_{i,2} + \cdots + E_{i,m}}{m}.$$

With the above, we define the cost function U to be the following long time limit:

$$\lim_{N\to\infty} \left\{ \sum_{i\in S} \frac{1}{2}\alpha_i \left(\frac{C_i(N)}{T(N)} - \Phi_i \right)^2 + \sum_{i\in S} \frac{1}{2}\beta_i \Big(\langle E_i(N) \rangle \Big)^2 \right\} \quad (1)$$

where the α_i's and β_i's are user-defined constants, and Φ_i's represent the prescribed distribution of per-PoI coverage times.

The intuition of the above cost function is as follows. The part with the α_i terms measures the discrepancy between the actual and prescribed distributions of the per-PoI coverage times. The part with the β_i terms measures the average per-PoI exposure time. Intuitively, minimizing the function U attempts to reduce both the coverage-time discrepancy and expected exposure time. However, the coverage-time and exposure-time measures are generally antagonistic so that the optimal choice of the transition probabilities is the result of a delicate trade-off between them. The parameters α_i's and β_i's reflect the relative importance between the two measures and they can be adjusted by the user.

Our next goal is to express the above cost function as an explicit function of the p_{ij}'s which completely characterize the Markov chain. For this, we will assume for the rest of the paper that the Markov chain is ergodic.

Let $\{\pi_i\}_{i\in S}$ be the stationary distribution of the Markov chain. By ergodicity, we have:

$$\lim_{N\to\infty} \tfrac{T(N)}{N} = \textstyle\sum_{j,k} \pi_j p_{jk} T_{jk}$$
$$\text{and } \lim_{N\to\infty} \tfrac{C_i(N)}{N} = \textstyle\sum_{j,k} \pi_j p_{jk} T_{jk,i}.$$

Hence the long term average of the coverage time distribution $\bar{C}_i$ can be defined and computed as:

$$\bar{C}_i = \lim_{N\to\infty} \frac{C_i(N)}{T(N)} = \frac{\sum_{j,k} \pi_j p_{jk} T_{jk,i}}{\sum_{j,k} \pi_j p_{jk} T_{jk}}. \quad (2)$$

In order to simplify the mathematical expression for the long time limit $\lim_{N\to\infty} \langle E_i(N) \rangle$, we will make the following assumptions which will not change our conclusions qualitatively:

- If a PoI is simply being passed by while the sensor is traveling between two other PoIs in a transition, then the passing-by is not considered a return visit.
- We assume that the time elapsed during each travel is always equal to one unit of time — this assumption is only enforced in the computation of the $\langle E_i \rangle$'s.

With the above simplifications, physically each exposure time segment $E_{i,j}$ is measured from the PoI location (e.g., j) immediately after the sensor has *left* i. Then necessarily, $j \neq i$. Let R_{ji} be the *expected value* of the time the sensor takes to reach (or return to) i from j. This quantity is also traditionally called the expected *first passage time*. As the probability of traveling from i to j is given by p_{ij}, we have the following expression for the long time average of the exposure times:

$$\bar{E}_i = \lim_{N\to\infty} \langle E_i(N) \rangle = \frac{\sum_{j\neq i} p_{ij} R_{ji}}{\sum_{j\neq i} p_{ij}} = \frac{\sum_{j\neq i} p_{ij} R_{ji}}{1 - p_{ii}}. \quad (3)$$

With the above, the cost function U can be written as:

$$\begin{aligned} U \;=\; & \sum_{i\in S} \frac{1}{2}\alpha_i \left[\sum_{j,k} \pi_j p_{jk} (T_{jk,i} - \Phi_i T_{jk}) \right]^2 \\ & + \sum_{i\in S} \frac{1}{2}\beta_i \left[\frac{\sum_{j\neq i} p_{ij} R_{ji}}{1 - p_{ii}} \right]^2. \end{aligned} \quad (4)$$

It is now time to remark about the search space of our optimization. In the current setting, we search within the space of *all* transition probabilities. This is in contrast to most problems in stochastic control, in which the search or *action space* is usually parameterized by a finite number of parameters. *Thus the novelty of our approach is that the solution gives the* true *optimal among the class of* all *Markov chains.* However, the solution space also becomes significantly more complex than less comprehensive formulations, and we will solve the problem by properly adding noise in the optimization procedure.

B. Analytical representation of U using generalized inverse

Even though the formula (4) for the cost function is highly explicit, it still requires the computation of the stationary distribution $\{\pi_i\}$ and the first passage time matrix $\{R_{ij}\}$. In order to compute them efficiently, we employ the concept of *generalized inverse*. In the following, we review the basic properties of this mathematical object, with particular application towards Markov transition matrices. The material is taken mainly from [16].

Let $P = \{p_{ij}\}_{i,j\in S}$ be an ergodic Markov transition matrix. Consider the generalized inverse $A^\sharp$ of $A = I - P$ which is defined as the matrix satisfying:

$$AA^\sharp A = A, \;\; A^\sharp A A^\sharp = A^\sharp, \;\text{ and }\; A^\sharp A = AA^\sharp.$$

The existence and computation of $A^\sharp$ is given by [16, Thm. 2.1, 5.1].

Using $A^\sharp$, the stationary distribution $\pi = \{\pi_i\}$ and the expected first passage time matrix $R = \{R_{ij}\}$ can be expressed as:

$$W = I - AA^\sharp \quad \text{([16, Thm. 2.3])} \quad (5)$$
$$R = (I - A^\sharp + JA^\sharp_{dg})D \quad \text{([16, Thm. 3.3, 3.6])} \quad (6)$$

where W is the matrix such that all of its rows are the stationary distribution π; J is the $M \times M$ square matrix with all of its entries equal to one; $A^\sharp_{dg}$ is the diagonal matrix consisting of the diagonal entries of $A^\sharp$; and D is the diagonal matrix such that $D_{ii} = \frac{1}{\pi_i}$. Note that we can have such a simple expression (6) for the R_{ij}'s is due to the simplifying assumptions we have for the exposure times.

The generalized inverse is also related to the commonly known *fundamental matrix* $Z = (I - P + W)^{-1}$ by the formula [16, Thm. 3.1]:

$$Z = I + PA^{\sharp} \tag{7}$$

Using Z, R can then be expressed as

$$R = (I - Z + JZ_{dg})D$$

$$\text{or component-wise:} \quad R_{ij} = \tfrac{\delta_{ij} - z_{ij} + z_{jj}}{\pi_j}. \tag{8}$$

The above result is the one used in the actual numerical computation.

Note that in principle the p_{ij}'s satisfy the constraints $0 \le p_{ij} \le 1$ and $\sum_{j \in S} p_{ij} = 1$, i.e., they lie in a higher dimensional polytope. Even though this property is manageable, to simplify the computational algorithm, we add a small penalization term to handle the case of $p_{ij} = 0$ or 1. More specifically, we consider the following penalized version U_ϵ of U:

$$\begin{aligned} U_\epsilon = & \sum_i \frac{1}{2}\alpha_i \left[\sum_{j,k} \pi_j p_{jk} \left(T_{jk,i} - \Phi_i T_{jk}\right)\right]^2 \\ & + \sum_i \frac{1}{2} \frac{\beta_i}{\pi_i^2 (1 - p_{ii})^2} \left[\sum_{j \ne i} p_{ij} (\delta_{ji} - z_{ji} + z_{ii})\right]^2 \\ & - \sum_{ij} \frac{1}{\epsilon} \left[\ln p_{ij}\right] (\epsilon - p_{ij})^2 \operatorname{sgn}(\epsilon - p_{ij}) \\ & - \sum_{ij} \frac{1}{\epsilon} \left[\ln(1 - p_{ij})\right] (1 - \epsilon - p_{ij})^2 \operatorname{sgn}(p_{ij} - 1 + \epsilon) \end{aligned} \tag{9}$$

where sgn(x) equals one for $x \ge 0$ and zero otherwise. Note that $U_\epsilon \longrightarrow +\infty$ as any of the p_{ij} converges to 0 or 1. This is prohibited by the steepest descent algorithm whose purpose is to minimize the U. The constraint $\sum_j p_{ij} = 1$ can be handled easily by a projection method, to be described later.

IV. Steepest Descent Optimization

This section describes the use of *steepest descent* to search for the minimum of U_ϵ. Let $U = U(q_1, q_2 \ldots q_l)$ be some function depending on the variables $q_1 \ldots q_l$. We would like to search for the minimum of U by evolving the variable q_i's. For simplicity, let $Q(t) = (q_1(t), \ldots q_l(t))$ denote the values of the q_i's at time t. Then, we compute:

$$\frac{d}{dt} U(Q(t)) = \nabla U(Q(t)) \frac{dQ(t)}{dt},$$

Now if we choose

$$\frac{dQ(t)}{dt} = -\nabla U(Q(t)),$$

then

$$\frac{d}{dt} U(Q(t)) = -\left|\nabla U(Q(t))\right|^2 < 0,$$

i.e. U is *decreasing*! Note that steepest descent as implemented above could only lead to a *local minimum* or *critical point*. In Section V, we will show how *stochastic perturbation* can solve this problem.

To apply the above formalism for the minimization of the cost function, we write U as:

$$U = U(\pi, Z, P) = U(\pi(P), Z(P), P) = U(P),$$

where $P = (p_{ij})$ is the transition matrix. Then

$$\begin{aligned} \frac{d}{dt} U\big(P(t)\big) = & \sum_i \frac{\partial U}{\partial \pi_i} \frac{d\pi_i(t)}{dt} + \sum_{jk} \frac{\partial U}{\partial z_{jk}} \frac{dz_{jk}(t)}{dt} \\ & + \sum_{jk} \frac{\partial U}{\partial p_{jk}} \frac{dp_{jk}(t)}{dt}. \end{aligned}$$

The expressions for $\frac{\partial U}{\partial \pi_i}$, $\frac{\partial U}{\partial z_{ij}}$ and $\frac{\partial U}{\partial p_{ij}}$ can be easily derived by straightforward partial differentiation. The formula for $\frac{d\pi_i(t)}{dt}$ and $\frac{dz_{jk}(t)}{dt}$ can be obtained by considering the *perturbation analysis* of Markov chain. For this, we follow [17] closely.

- By [17, Formula (23c), (15), and (17)]:
  $\frac{d\pi}{dt} = \pi \dot{P} Z$, or component-wise, we have
  $$\tfrac{d\pi_i}{dt} = \textstyle\sum_{k,j} \pi_k z_{ji} \dot{P}_{kj}.$$
- By [17, Formula (23d), (16), and (18)]:
  $\frac{dZ}{dt} = Z\dot{P}Z - W\dot{P}(Z^2)$ or component-wise,
  $$\tfrac{dz_{ij}}{dt} = \textstyle\sum_{kl} \left[z_{ik} z_{lj} - \pi_k (Z^2)_{lj}\right] \dot{P}_{kl}$$
  where $(Z^2)_{lj} = \sum_m z_{lm} z_{mj}$.

Using the above, we get that $\frac{dU}{dt}$ equals

$$\begin{aligned} & \sum_i \frac{\partial U}{\partial \pi_i} \frac{d\pi_i}{dt} + \sum_{ij} \frac{\partial U}{\partial z_{ij}} \frac{dz_{ij}}{dt} + \sum_{ij} \frac{\partial U}{\partial p_{ij}} \frac{dp_{ij}}{dt} \\ = & \sum_i \frac{\partial U}{\partial \pi_i} \left[\sum_{k,l} \pi_k z_{li} \dot{P}_{kl}\right] \\ & + \sum_{ij} \frac{\partial U}{\partial z_{ij}} \left[\sum_{kl} \left[z_{ik} z_{lj} - \pi_k (Z^2)_{lj}\right] \dot{P}_{kl}\right] + \sum_{ij} \frac{\partial U}{\partial p_{ij}} \frac{dp_{ij}}{dt} \\ = & \sum_{kl} \left\{ \left[\sum_i \pi_k z_{li} \frac{\partial U}{\partial \pi_i}\right] + \left[\sum_{ij} \frac{\partial U}{\partial z_{ij}} \left[z_{ik} z_{lj} - \pi_k (Z^2)_{lj}\right]\right] \right. \\ & \left. + \frac{\partial U}{\partial p_{kl}} \right\} \dot{P}_{kl}. \end{aligned}$$

If we let $[D_P U]_{kl}$ be the expression

$$\left[\sum_i \pi_k z_{li} \frac{\partial U}{\partial \pi_i}\right] + \left[\sum_{ij} \frac{\partial U}{\partial z_{ij}} \left[z_{ik} z_{lj} - \pi_k (Z^2)_{lj}\right]\right] + \frac{\partial U}{\partial p_{kl}}, \tag{10}$$

then we have:

$$\frac{dU}{dt} = \sum_{kl} [D_P U]_{kl} \dot{P}_{kl} = \left\langle D_P U, \dot{P} \right\rangle.$$

We can simply take: $\dot{P}_{kl} = -[D_P U]_{kl}$. However, as mentioned earlier, $P(t)$ must be a transition matrix at all t, i.e. it has to satisfy for all j, that $\sum_k p_{jk}(t) = 1$. To accommodate this, we *project orthogonally* $D_P U$ onto this linear subspace, i.e.,

$$\frac{dU}{dt} = \left\langle \Pi[D_P U], \dot{P} \right\rangle$$

and then take:

$$\dot{P}_{kl} = -\left(\Pi[D_P U]\right)_{kl}.$$

The formula for the projection Π is given by:

$$\Pi_{ij} = U_{ij} - \frac{\sum_k U_{ik}}{M}, \quad \text{for all } i, j\ , \tag{11}$$

where U_{ij} and Π_{ij} are the entries of $[D_P U]$ and $\Pi[D_P U]$, respectively. Note that $\sum_j \Pi_{ij} = 0$, i.e., the sum of each row of Π is *zero*. Hence, the property that the quantities represent a transition probability matrix is preserved at all time.

V. Computational Algorithm

The above described steepest descent algorithm will be implemented as follows:

1) Start with an arbitrary ergodic transition probability P.
2) Compute $[D_P U]$ (10) and its projection $\Pi[D_P U]$ (11).
3) Set $V = -\Pi[D_P U]$.
4) Set the new P as $P + V * \triangle t$, where $\triangle t$ is some small time step.
5) For the new P, compute π, Z and R by (5), (7) and (8).
6) Go back to step two, or stop if the optimal is attained (within some tolerance level).

Note that the ergodicity of P is ensured by the finiteness of all the R_{ij}'s which is indeed preserved during the whole computation.

In the rest of the paper, we will study the following variants of the steepest descent algorithm:

V1: Basic algorithm. All the p_{ij} values are initially set to $\frac{1}{M}$, where M is the number of PoIs. A constant time step Δt is used. This provides a basic test for the validity of our steepest descent algorithm.

V2: Random initial data. In this case, the initial values of the p_{ij}'s are random. They are set to $\frac{rand \times rem}{M}$ except for the entries in the last column, which will be set to rem, where $rand$ is a random number between 0 and 1, rem is the remaining probability for p_{ij}'s within a row (as the summation of a row should be one), and M is the number of PoIs. This test will ensure that our algorithm is stable with respect to the initial condition.

V3: Adaptive time step Δt. With the gradient information ∇U in the current transition matrix P, the optimal time step Δt_* is chosen as follows:

$$\Delta t_* = \min_{\delta > 0} U(P - \delta \nabla U(P)).$$

The boundaries of δ are first determined with respect to the constraint that $0 \leq p_{ij} \leq 1$ after the update. As it is unclear how the utility changes within the range of δ, a conservative *trisection method* (wherein only one sub-section is removed as we tighten the bounds) is used to tighten the range until Δt_* is found. The algorithm terminates when $\Delta t_* = 0$. We define the terminating point as a *local optimum* when in fact there exists another terminating point with a lower cost.

V4: Stochastic perturbations. This step is to make sure that our algorithm will not get stuck at a local minimum of the cost function, which is shown to be essential in the next section. The $[D_P U]$ values are perturbed by mean zero Gaussian noise with standard deviation σ. Δt_* is then computed using the perturbed $[D_P U]$. If $\Delta t_* = 0$, then $\Delta t = rand$, where $rand$ is a random number between the boundaries of Δt, which are determined with respect to the constraint that $0 \leq p_{ij} \leq 1$ after the update. The updates to p_{ij}'s are accepted if the computed utility is better than the original values; otherwise, they are accepted with probability $e^{\frac{-\Delta_U}{k \times \log(count)}}$, where Δ_U is the worsening in cost U after normalization by the *best cost* computed so far, $count$ is the number of iterations already performed by the algorithm, and k is a constant parameter. The normalization is important because it allows us to determine how high the algorithm should "jump" (in trying to get out of a local optimum) without knowing the range of the cost function U_ϵ (which is frequently unknown beforehand). Intuitively, the algorithm accepts unattractive changes with higher probability at the beginning of the search, and this probability decreases as the algorithm proceeds. This way of accepting the updates is similar to the cooling process in simulated annealing. The optimality and convergence of simulated annealing has a strong theoretical basis [18]. However, the proofs are mostly of theoretical interest, since the convergence in practice is usually much faster, as we show in Section VI-C.

Henceforth in the paper, we will use *steepest descent algorithm* to refer to our general solution approach. Additionally, we will call variant V1 of the algorithm the *basic algorithm*, the basic algorithm modified by variants V2 and V3 the *adaptive algorithm*, and the basic algorithm modified by all of variants V2–V4 the *stochastically perturbed algorithm* or simply *perturbed algorithm*. We have verified the performance of the steepest descent by using the transition matrices computed by the algorithm at each iteration to drive a corresponding simulation of the mobile sensor's coverage. We found that the algorithm is able to achieve good tradeoffs between the coverage-time and exposure-time metrics, as the weightings of the two parameters in the cost function are varied. Also, the adaptive algorithm can speed up convergence to the final solution, and the perturbed algorithm can additionally converge to the globally optimal steady state in practically all of the scenarios.

VI. Performance measurements

In this section we study the performance of the steepest descent algorithm. For simplicity of exposition, we consider the case that the α_i's and β_i's in Eq. (9) are all equal, i.e., $\alpha_1 = \alpha_2 = \cdots = \alpha_M = \alpha$ and $\beta_1 = \beta_2 = \cdots = \beta_M = \beta$. We further define the coverage-time deviation ΔC as

$$\Delta C = \sum_{i \in S} \left[\sum_{j,k} \pi_j p_{jk} (T_{jk,i} - \Phi_i T_{jk}) \right]^2 \tag{12}$$

and the average exposure time $\bar{E}$ as

$$\bar{E} = \sqrt{\sum_{i \in S} \left[\frac{\sum_{j \neq i} p_{ij} R_{ji}}{1 - p_{ii}} \right]^2}. \tag{13}$$

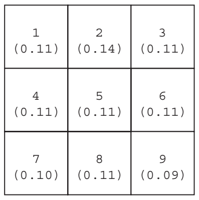
Hence, the cost function in Eq. (4) can be rewritten as

$$U = \frac{1}{2} \alpha \Delta C + \frac{1}{2} \beta \bar{E}^2. \tag{14}$$

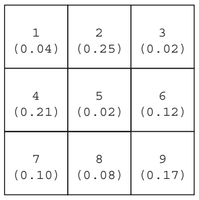
Note that (i) we use the ΔC defined above instead of $\sum_i (\bar{C}_i - \Phi_i)^2$ (see Eq. (2)) as it has an easier computational expression. Qualitatively, both of them measure the discrepancy between the actual and desired coverage profiles; (ii) the quantity $\bar{E}$ is related to the long time average exposure times as $\bar{E} = \sqrt{\sum_i \bar{E}_i^2}$ (see Eq. (3)).

We will study the following performance aspects of the steepest descent algorithm.

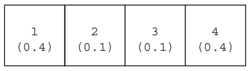
Trade-off between coverage-time and exposure-time metrics. We examine the characteristics of the transition



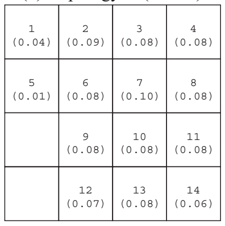
(a) Topology 1 (3×3)



(b) Topology 2 (3×3)

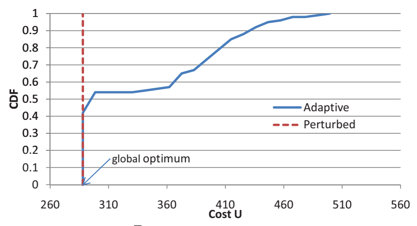


(c) Topology 3 (1×4)

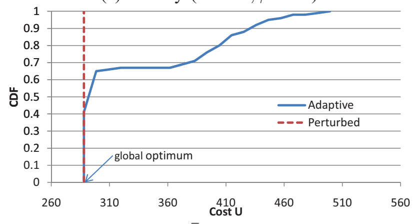


(d) Topology 4 (4×4)

Figure 1. Target per-PoI shares of coverage times Φ_i in four simulation topologies.



(a) $\bar{E}$ only ($\alpha = 0, \beta = 1$)



(b) ΔC and $\bar{E}$ ($\alpha = 1, \beta = 1$)

Figure 2. CDFs of achieved cost U_ϵ in a large number of runs by the adaptive algorithm and perturbed algorithm **(Topology 1)**.

matrix and the steady state distribution of the sensor's locations, as we vary the weight of the coverage-time deviation ΔC (Eq. (12)) and that of the average exposure time $\bar{E}$ (Eq. (13)), i.e., α and β respectively in Eq. (14).

Stability and adaptivity of the algorithm. We evaluate the cost U (Eq. (14)) of the transition matrix in terms of the *iteration number* of the steepest descent algorithm. (Recall that the algorithm works by iteratively updating the transition matrix.) We also compare the U value achieved by the basic algorithm and its variants given in Section V, and test if the algorithm gets stuck at a local minimum.

Performance of actual simulation of the coverage schedules. We verify the performance of steepest descent by applying the computed transition matrices to control the movement of the mobile sensor in the Markov chain simulations. The sensor in a simulation picks the destination of its next transition according to the transition matrix, and pauses at the destination for a fixed time interval before its next transition. We measure ΔC in Eq. (12) and $\bar{E}$ in Eq. (13). Since the simulation uses a stabilized U value as stopping criterion, we verify that $\bar{E}$ has also stabilized when the simulation ends.

In our experiments, we use the four topologies shown in Fig. 1. Each PoI i in the topologies is the center of the cell labeled with i, and its targeted share of coverage time Φ_i is in parentheses. We assume that in traveling from i to j, the sensor uses the straight-line path between i and j.

A. Existence of numerous local optima

We characterize the search space by comparing the adaptive algorithm (i.e., the basic algorithm modified by variants V2 and V3 in Section V) and the perturbed algorithm (i.e., the basic algorithm modified by variants V2–V4 in Section V) under Topology 1 in Fig. 1(a). In computing the cost function, we illustrate the cases of considering (i) $\bar{E}$ only ($\alpha = 0, \beta = 1$) and (ii) both ΔC and $\bar{E}$ ($\alpha = 1, \beta = 1$), while having $\epsilon = 0.0001$ and $k = 10000$. The CDFs of the achieved costs U_ϵ by the adaptive and perturbed algorithms are shown in Fig. 2(a) and 2(b) for the cases (i) and (ii), respectively.

	$\bar{C}_i$			
$\alpha : \beta$	1	2	3	4
0:1	0.214	0.286	0.286	0.214
1:1	0.250	0.250	0.250	0.250
1:0.01	0.254	0.246	0.246	0.254
1:0.0001	0.369	0.130	0.131	0.370
1:0.000001	0.399	0.101	0.101	0.399
1:0	0.4	0.1	0.1	0.4

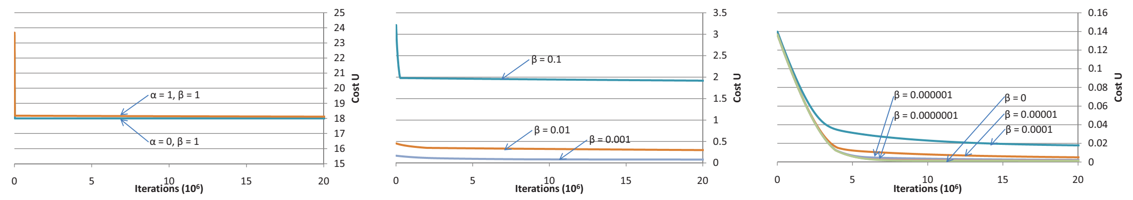
Table I
$\bar{C}_i$ FOR $\epsilon = 0.0001$, $\Delta t = 0.000001$.

From the figures, observe that the adaptive algorithm hits a large number of local optima in its search, from which it is not able to discover a better cost U in the computed descent direction. The presence of numerous local optima makes it essential to adopt noise in the search process, as detailed by the design of the perturbed algorithm in Section V. From Fig. 2, notice that the perturbed algorithm achieves much better performance than the adaptive algorithm, as evidenced by the extremely sharp rise of the CDF at the global optimum solution. *Indeed, the perturbed algorithm computes a solution extremely close to, if not exactly the same as, the global optimum in all the runs of the experiment, irrespective of the initial search parameters.*

B. Trade-off between ΔC and $\bar{E}$ metrics

We study the characteristics of the transition matrix $\{p_{ij}\}$ and the steady-state distribution π_i of the Markov chain, when we vary α and β in Eq. (14), using Topology 3 in Fig. 1(c). We use $\epsilon = 0.0001$ and $\Delta t = 0.000001$, and vary β from 1 to 0.0000001 while keeping $\alpha = 1$. We also study two extreme cases: (i) $\alpha = 1$ and $\beta = 0$ (we are concerned with the ΔC metric only); and (ii) $\alpha = 0$ and $\beta = 1$ (we are concerned with the $\bar{E}$ metric only). The results for the $\bar{C}_i$ and $\bar{E}_i$ (defined in Eq. (2) and (3)) are shown in Tables I and II, respectively.

Observations. From the simulation results, we deduce that hen we reduce β, $\bar{C}_i$ (the fraction of time PoI i is actually covered) will more closely approximate the target share of coverage time, while $\bar{E}_i$ (the average exposure time of i) grows, as shown in Tables I and II. Hence, ΔC decreases while $\bar{E}$ increases. It is because upon reducing β, we are less



(a) $\beta = 1$ (b) $\alpha = 1$ (c) $\alpha = 1$

Figure 3. Cost function value U in terms of the iteration number. $\epsilon = 0.0001$, $\Delta t = 0.000001$.

$\alpha : \beta$	E_i			
	1	2	3	4
0:1	9.001	9.001	9.001	9.001
1:1	8.992	9.040	9.004	8.960
1:0.01	8.255	9.836	9.835	8.254
1:0.0001	29.770	62.513	62.099	29.524
1:0.000001	48.264	118.937	118.898	48.290
1:0	6.944	13742.767	13742.768	6.944

Table II

$\bar{E}_i$ FOR $\epsilon = 0.0001$, $\Delta t = 0.000001$.

Algorithm	min U	max U	avg U
Adaptive	288.0125	499.3528	332.3202
Perturbed	288.0125	288.0145	288.0127

Table III

THE MINIMUM, MAXIMUM, AND AVERAGE OPTIMAL COST DETERMINED BY THE ADAPTIVE ALGORITHM AND THE PERTURBED ALGORITHM (**$\alpha = 0$, $\beta = 1$, Topology 1**).

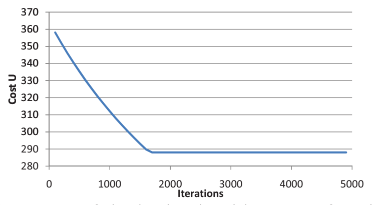


Figure 4. Performance of the basic algorithm: cost function value U as a function of iteration number (**$\alpha = 0$, $\beta = 1$, Topology 1**).

concerned about the exposure time of the PoI, and are more focused on the target coverage time. We can also observe that $\bar{C}_i$ and $\bar{E}_i$ are not significantly changed when β is large. It is because for Topology 3, the magnitude of $\bar{E}$ is significantly larger than ΔC. Hence, even if β (weight of $\bar{E}$) is reduced from 1 to 0.01, the exposure-time component still dominates the cost function, and the resulting p_{ij} and π_i do not change significantly. We notice that as the size of the topology grows, $\bar{E}$ grows, and we will need to further reduce β before the coverage-time component in U will have an observable effect on the p_{ij} and π_i.

C. Adaptivity and stability of the algorithm.

In this set of experiments, we trace the evolution of the cost function U as the transition matrix changes in each iteration of the steepest descent algorithm. We verify that steepest descent can progressively improve the transition matrix by reducing U over the iterations. To demonstrate that the algorithm does not get stuck at a local minimum, we show convergence to the same transition matrix and U value from different initial p_{ij}'s using the perturbed algorithm in Section V.

We study the stability and adaptivity of the algorithm under three settings ($\epsilon = 0.0001, \ \Delta t = 0.000001$):

ΔC only ($\alpha = 1, \ \beta = 0$). We evaluate the case when the cost function considers the coverage-time deviation metric only. The results of the basic algorithm using Topology 2 are depicted in Fig. 5(a). In addition, we evaluate the perturbed algorithm with different initial values of p_{ij}'s (i.e., the initial p_{ij}'s are generated using different random seeds), and the results are shown in Fig. 5(b).

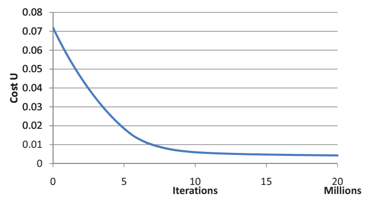
$\bar{E}$ only ($\alpha = 0, \ \beta = 1$). We evaluate the case when the cost function considers the exposure-time metric only. The results of the basic algorithm are depicted in Fig. 4. In addition, we evaluate the cases when the adaptive algorithm and the perturbed algorithm (recall the definitions in Section V) are used. The minimum, maximum, and average optimal cost determined by the algorithms using Topology 1 of 200 independent runs are summarized in Table III.

Both ΔC and $\bar{E}$ ($\alpha = 1, \ \beta = 1$). We further consider the case the coverage-time deviation metric and the exposure time metric have the same weight (i.e., $\alpha = \beta$). We omit plots of results for the basic algorithm because they are quite similar to the case when we consider $\bar{E}$ only. The results for different α and β values for the basic algorithm using Topology 3 are shown in Fig. 3.

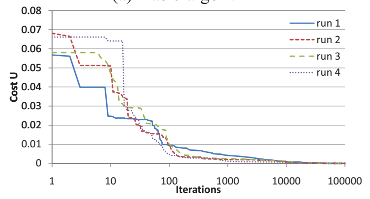
Observations. From the simulation results, we deduce that: (1) During the optimization process, the cost function generally decreases until it reaches a stable value as depicted in Fig. 3 and 5. However, the marginal reduction in the cost becomes smaller as the number of iterations becomes larger.
(2) Using different random seeds to generate the initial p_{ij}'s does not affect the performance of the perturbed algorithm (Fig. 5(b)) in that it will converge to the same stable values of the cost function. However, the convergence process and its time will be affected. Moreover, the perturbed algorithm can converge extremely close to the same optimal costs in all these situations, i.e., the algorithm is not trapped at a local minimum. This result also holds for more complex topologies.
(3) The best solution by the adaptive algorithm (i.e., without noise in the optimization) can have a large range depending on where the search starts. For instance, Table III shows that the difference between the minimum and maximum returned best costs by the adaptive algorithm is much larger than that by the perturbed algorithm. The average performance of the perturbed algorithm is also much better than that of the adaptive algorithm. The difference in performance is due to the adaptive algorithm being trapped at one of the numerous local optima in the solution space, and the perturbed algorithm's ability to jump out of the local optima.

D. Performance of actual Markov chain simulations

We evaluate the performance of actual sensor schedules as they are controlled by the Markov chain with the transition

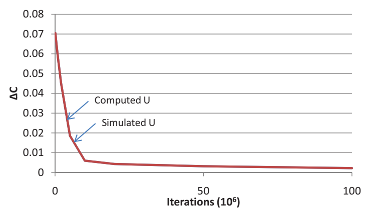


(a) Basic algorithm

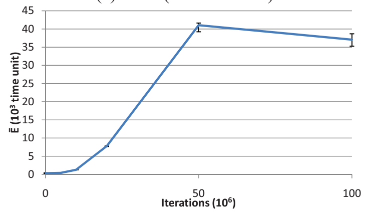


(b) Perturbed algorithm with different initial p_{ij}'s

Figure 5. Performance of (a) the basic algorithm, and (b) the perturbed algorithm with different initial p_{ij}'s: cost function value U as a function of the iteration number **($\alpha = 1$, $\beta = 0$, Topology 2)**.

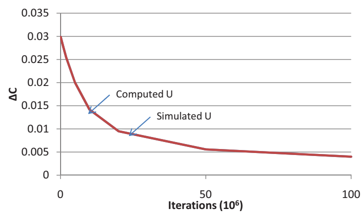


(a) ΔC (cost function)

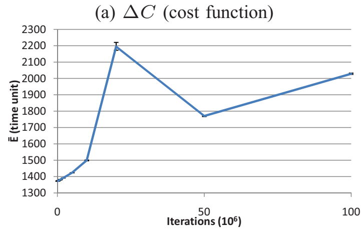


(b) $\bar{E}$

Figure 6. Performance of coverage algorithm: (a) ΔC and (b) $\bar{E}$ as a function of the iteration number **($\alpha = 1$, $\beta = 0$, Topology 2)**.

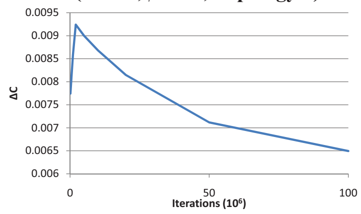


(a) ΔC (cost function)

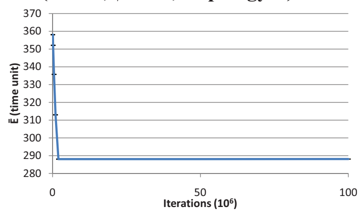


(b) $\bar{E}$

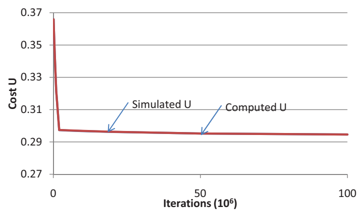
Figure 7. Performance of algorithm: (a) ΔC and (b) $\bar{E}$ as a function of the iteration number **($\alpha = 1$, $\beta = 0$, Topology 4)**.



(a) ΔC



(b) $\bar{E}$



(c) Overall cost function U

Figure 8. Performance of algorithm: (a) ΔC, (b) $\bar{E}$, and (c) overall cost function U as a function of iteration number **($\alpha = 1$, $\beta = 0.0001$, Topology 1)**.

$\alpha : \beta$	ΔC	E
0:1	0.0095	288.002
1:1	0.0072	288.010
1:0.0001	0.0052	288.974
1:0	0.0012	1143.45

Table IV
PERFORMANCE FOR DIFFERENT α/β RATIOS USING TOPOLOGY 1.

matrices found by the steepest descent algorithm. We verify that the cost function predicted by steepest descent indeed reflects the realized ΔC and $\bar{E}$ metrics when the computed transition probabilities are applied in practice.

We vary the weights α and β in Eq. (9) as before, and use steepest descent to compute the optimal transition matrix. A matrix generated by each iteration of the steepest descent algorithm is used to drive a corresponding Markov chain simulation of the mobile sensor schedule, and the ΔC and $\bar{E}$ values are measured. Each simulation is repeated ten times to obtain the reported average. 25-th and 75-th percentiles of the measured values are reported as error bars where they are significant. Combinations of the α and β used in different cases are as follows (again $\epsilon = 0.0001$, $\Delta t = 0.000001$):

ΔC only ($\alpha = 1$, $\beta = 0$). The cost function considers only the coverage-time deviation measure. The results are shown in Fig. 6 and 7 for Topology 2 and Topology 4, respectively.

$\bar{E}$ only ($\alpha = 0$, $\beta = 1$). The cost function considers the exposure-time metric only. As the steepest descent algorithm stabilizes very quickly, we only evaluate the stabilized transition matrix in the sensor simulations, and the results are shown in Table IV.

Both ΔC and $\bar{E}$ ($\alpha = 1$, $\beta = 0.0001$). We study the case in which we use a small β. The results are shown in Fig. 8 for Topology 1.

Observations. From the simulation results, we deduce that (1) When $\beta = 0$, the measured U in the simulations gives a perfect match with the U value computed by steepest descent (Fig. 6 and 7). If exposure time is also considered ($\beta > 0$), the measured U in the simulations gives a very close match with the computed U by the steepest descent (Fig. 8). However, the match is not exact because in computing $\bar{E}$, the analytical formula makes the simplifying assumption that the time duration of each transition is the same, which does not hold in practice. However, observe from Fig. 8 that the percentage difference between the simulated and computed numbers is very small.
(2) If $\beta = 0$, then $\bar{E}$ becomes big, but it does not necessarily worsen as the ΔC metric improves (Fig. 6). Moreover, the extent to which $\bar{E}$ worsens is not determined by the size of the map, but the target allocation of per-PoI coverage times (compare Fig. 6 and 7).

VII. CONCLUSION AND DISCUSSIONS

As discussed, the main contribution of our work is the computational framework for the search of the optimal transition probabilities of a Markov chain so as to minimize some cost function (or equivalently, the optimization of some utility function) with application to the scheduling of mobile sensor coverage. The key feature of the cost function is that it contains *conflicting* or *antagonistic* criteria such as the coverage and exposure time metrics. The novelty of our method of solution, by means of *steepest descent*, is its *simplicity* and the *comprehensiveness* of the search space – we look for optimal transition rates in the space of *all* transition probability matrices. Underlying this approach is the explicit expression of the cost (or utility) function in terms of the transition probabilities. The further advantage of our approach is the ease for individual users to adjust the relative importance of the various criteria suitable for different applications. A challenge of the comprehensive optimization, however, is the complexity of the solution space empirically evidenced by a large number of local optima. We overcome the challenge successfully by incorporating noise in the optimization.

Consideration for other physical concepts in the optimization can be addressed as follows:

Energy cost. The cost of energy can also be taken into account. For example, assuming that energy use is proportional to the distance traveled, we can consider the following expression:

$$D = \sum_i \pi_i \left[\sum_{j \neq i} p_{ij} d_{ij} \right],$$

where d_{ij} indicates the travel cost from PoI i to j (for $j \neq i$). The factor π_i represents the stationary distribution of the sensor locations in the long run. We can consider adding (1) D^2 into the cost function so as to control the total cost; or (2) $(D - \gamma)^2$ where γ is some user-defined constant, which leads to the effect that on average, it is *required* or *advantageous* to have some *prescribed* sensor movement per unit time.

Entropy of Markov chain. One of the main advantages of introducing randomness into the mobile sensor scheduling is its *unpredictability*. Hence it is natural to search for the transition rates with *maximum* entropy. For Markov chain, this concept can be quantitatively measured by (see for example [19, p. 76]):

$$H = -\sum_i \pi_i \left[\sum_j p_{ij} \ln p_{ij} \right].$$

We can consider a new cost function such as $U - H$ so as to minimize the *combined* cost and entropy. Another alternative is $U - \epsilon H$, where ϵ is a small parameter. This has similar effects on minimizing U with maximum entropy.

Acknowledgment

Research was supported in part by U.S. DOE SensorNet grant no. AC05-00OR22725 and Mathematics of Complex, Distributed, and Interconnected Systems Program, Office of Advanced Computing Research; NSF DMS-0707926; NSFC 60974122; 111 Projects B07031; SKLICT Open Funding ICT0901; and SFCU 2009QNA5007.

References

[1] P. McKenney, "Stochastic Fairness Queueing," in *Proc. of IEEE INFOCOM*, March 1990.

[2] A. K. Parekh and R. G. Gallager, "A generalized processor sharing approach to flow control in integrated services networks: The single node case," *IEEE/ACM Transactions on Networking*, vol. 1, June 1993.

[3] C. A. Waldspurger and W. E. Weihl, "Lottery scheduling: flexible proportional-share resource management," in *Proc. of the First Symposium on Operating System Design and Implementation*, 1994.

[4] C. A. Waldspurger, "Lottery and stride scheduling: Flexible proportional-share resource management," Ph.D. dissertation, Massachusetts Institute of Technology, September 1995.

[5] J. C. R. Bennett and H. Zhang, "WF^{2}Q: Worst-case fair weighted fair queueing," in *Proc. of IEEE INFOCOM*, 1996.

[6] N. Bisnik, A. Abouzeid, and V. Isler, "Stochastic event capture using mobile sensors subject to a quality metric," in *Proc. of MOBICOM*, September 2006.

[7] Z. Butler and D. Rus, "Event-based motion control for mobile sensor networks," *IEEE Pervasive Computing*, vol. 2, 2003.

[8] M. Tariq, M. Ammar, and E. Zegura, "Message Ferry Route Design for Sparse Ad hoc Networks with Mobile Nodes," in *Proc. of MobiHoc*, Florence, Italy, May 2006.

[9] P. Ogren, E. Fiorelli, and N. Leonard, "Cooperative control of mobile sensor networks: Adaptive gradient climbing in a distributed environment," *IEEE Trans. Automatic Control*, vol. 49, 2004.

[10] S. A. McKenna, D. B. Hart, and L. Yarrington, "Impact of sensor performance on protecting water distribution systems from contamination events," *ASCE Journal of Water Resources Planning and Management*, vol. 134, 2006.

[11] R. Murray, R. Janke, W. E. Hart, J. W. Berry, T. Taxon, and J. Uber, "Sensor network design of contamination warning systems: A decision framework," *e-Journal AWWA*, vol. 100, 2008.

[12] A. Ostfeld, J. G. Uber, and E. Salomons, "Battle of water sensor networks: A design challenge for engineers and algorithms," in *In 8th Symposium on Water Distribution Systems Analysis*, 2006.

[13] W. K. Hastings, "Monte carlo sampling methods using markov chains and their applications," *Biometrika*, vol. 57, 1970.

[14] N. A. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, "Equation of state calculations by fast computing machines," *The Journal of. Chemical Physics*, vol. 21, 1953.

[15] R. L. Cruz, "Quality of service guarantees in virtual circuit switched networks," *IEEE Journal on Selected Areas in Communications*, vol. 13, 1995.

[16] C. D. Meyer, "The role of the group generalized inverse in the theory of finite markov chains," *SIAM Review*, vol. 17, 1975.

[17] P. J. Schweitzer, "Perturbation theory and finite markov chains," *J. Appl. Prob.*, vol. 5, 1968.

[18] B. Hajek, "Cooling schedules for optimal annealing," *Mathematics of Operations Research*, vol. 13, no. 2, May 1988.

[19] L. Koralov and Y. G. Sinai, *Theory of Probability and Random Processes*. Springer-Verlag, 2007.