

MIT Open Access Articles

Correlated Orienteering Problem and its application to informative path planning for persistent monitoring tasks

The MIT Faculty has made this article openly available. **Please share** how this access benefits you. Your story matters.

Citation: Yu, Jingjin, Schwager, Mac and Rus, Daniela. 2014. "Correlated Orienteering Problem and its application to informative path planning for persistent monitoring tasks."

As Published: 10.1109/iros.2014.6942582

Publisher: IEEE

Persistent URL: <https://hdl.handle.net/1721.1/137110>

Version: Author's final manuscript: final author's manuscript post peer review, without publisher's formatting or copy editing

Terms of use: Creative Commons Attribution-Noncommercial-Share Alike



Correlated Orienteering Problem and its Application to Persistent Monitoring Tasks

Jingjin Yu Mac Schwager Daniela Rus

Abstract

We propose a novel non-linear extension to the Orienteering Problem (OP), called the Correlated Orienteering Problem (COP). We use COP to model the planning of informative tours (cyclic paths) for the persistent monitoring of a spatiotemporal field with time-invariant spatial correlations, in which the tours are constrained to have fixed-length or -time budgets. Our focus in this paper is QCOP, a quadratic COP formulation that only looks at correlations between neighboring nodes in a node network. The main feature of QCOP is a quadratic utility function that captures the said spatial correlation. QCOP may be solved using mixed integer quadratic programming (MIQP), with the resulting anytime algorithm capable of planning multiple disjoint tours that maximize the quadratic utility. In particular, our algorithm can quickly plan a near-optimal tour over a network with up to 150 nodes. Besides performing extensive simulation studies to verify the algorithm's correctness and characterize its performance, we also successfully applied it to two realistic persistent monitoring tasks: (i) estimation over a synthetic spatiotemporal field, and (ii) estimating the temperature distribution in the state of Massachusetts.

I. INTRODUCTION

Consider the problem of dispatching unmanned aerial vehicles (UAVs) with on-board cameras to monitor road traffic in a large city. Often, UAVs have limited range and can stay in air only for a limited amount of time. On the other hand, traffic events such as congestion tend to have strong local correlations, *i.e.*, if the vehicle density at an intersection is high, the same is likely true at intersections that are close-by. Therefore, sequentially visiting intersections following the road network's topological structure may offer little incremental information. As UAVs are not restricted to travel along roads, routes with carefully selected, not necessarily adjacent intersections can potentially offer much better overall traffic information per unit of traveled distance. Under such settings, the following question then naturally arises: how to plan the best tours for the UAVs so that they can collect the maximum amount of traffic information per flight? Application scenarios like this are far from unique. For example, nearly identical settings appear when we want to deploy autonomous marine vehicles to collect samples for the detection of water pollution events such as oil spills, or when a political candidate wants to maximize his/her reach given limited travel and time budgets. A graphical example illustrating the settings of such problems is provided in Figure 1.

In persistent surveillance and monitoring tasks using mobile robots with onboard sensors, the robots usually have fixed base stations which they must depart from and return to. Moreover, these robots often have limited travel distance or time budget. Thus, when a large number of points of interest (nodes) must be surveyed, it may well be the case that only a subset of the nodes can be visited by the robots. Then, choices among the nodes must be made to accommodate two conflicting goals: (i) each robot must follow a tour (cyclic path) of which the total cost does not exceed its travel budget, and (ii) the tours must be planned to maximize the amount of collected information, as measured by some utility (reward) function. When nodes have information utilities that are additive, an *Orienteering Problem* (OP) Vansteenwegen et al. (2011), a problem closely related to the well known *Traveling Salesman Problem* (TSP) Laporte (1992), is obtained. Research on OP has yielded effective algorithms for solving many versions of the problem, including the *Team Orienteering Problem* (TOP) Chao et al. (1996a), in which multiple tours must be planned.¹

J. Yu and D. Rus with the Computer Science and Artificial Intelligence Lab at the Massachusetts Institute of Technology. E-mail: jingjin.rus@csail.mit.edu.

M. Schwager is with the Mechanical Engineering Department at Boston University. E-mail: schwager@bu.edu.

This work was supported in part by ONR projects N00014-12-1-1000 and N00014-09-1-1051.

¹Henceforth, we use *Orienteering Problem* (OP) as a blanket term to cover all additive/linear utility orienteering problems, which include TOP.

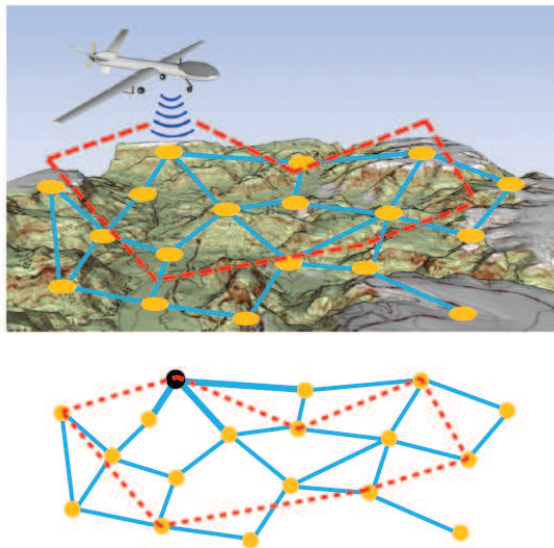


Fig. 1. [top] A surveillance scenario in which an UAV with limited range is faced with the problem of covering a large number of nodes. [bottom] The abstracted node network (dots and solid lines) and the tour (dashed lines) taken by the UAV. As a measurement is made from the UAV at a node, for example the dark node in the figure, through spatial correlation, partial information about its neighbors can also be inferred from the same measurement (in this case, the neighbors are the three nodes connected to the dark node though the three bold edges). Following the dashed tour lines, which is much shorter than a traveling salesman (TSP) tour, the UAV can provide at least partial information about every node in the network.

In practice, however, the information collected at a node is frequently correlated with the information collected at adjacent nodes, rendering the total utility a non-linear combination of individual node utility. That is, it is often the case that such information can be viewed as forming a locally-correlated spatiotemporal field; surveying a given node will also offer partial information about its neighbors. For example, the nodes may be cities, city blocks, and locations in reservoirs with the associated quantities being population dynamics, criminal activities, and water pollutant concentration, respectively. In this paper, assuming that the spatial correlation among the nodes are intrinsic (*i.e.*, determined by local structures and mostly time-invariant) we propose an extension to OP, called the *Correlated Orienteering Problem (COP)*, to incorporate such correlations in the informative path planning phase. We do not assume convexity or submodularity over the underlying field. In particular, we focus on QCOP, a COP instantiation that only looks at correlations between immediate neighbors. After formulating COP and QCOP, we derive mixed integer quadratic programming (MIQP) models for solving the problem for single and multiple robots. Our simulation studies suggest that (i) QCOP captures spatial correlations among the nodes quite well, and (ii) the MIQP-based *anytime* algorithm quickly yields approximate solutions to the QCOP problem for instances of up to 150 nodes with user-specified optimality bounds. We note that COP and QCOP are NP-hard problems.

Related work: Our work fuses ideas from two relatively disjoint branches of research: (discrete) OP and (mostly continuous) informative path and policy planning problems. OP, as indicated by its name, has its origin from orienteering games Chao et al. (1996a,b). In such a game, rewards of uniform or varying sizes are spatially scattered. To collect a reward, a player must physically visit the location where the reward is placed to pick it up. The goal for a player or a team of players is to plan the best path(s) to gather the maximum possible reward given limited time. Thus, OP can be viewed as a variation of both the Knapsack Problem (KP) Karp (1972) and the Traveling Salesman Problem (TSP) Laporte (1992). For a detailed account of OP, see Vansteenwegen et al. (2011).

The literature on informative path and policy planning for persistent monitoring is fairly rich Michael et al. (2011); Smith et al. (2011); Alamdari et al. (2012); Arvelo et al. (2012); Cassandras et al. (2013); Girard et al. (2004); Grocholsky et al. (2006); Lan and Schwager (2013); Nigam and Kroo (2008); Ny et al. (2008); Smith et al. (2012); Soltero et al. (2012); Hollinger and Sukhatme (2014); Yu et al. (2014a), covering theories, systems, algorithm designs, and applications. In the works presented in Alamdari et al. (2012); Arvelo et al. (2012); Lan and Schwager (2013); Ny et al. (2008); Smith et al. (2012); Hollinger and Sukhatme (2014), fundamental limits as well as provably correct algorithms were established for a variety of persistent monitoring problems. At the same time,

comprehensive systems have been designed to address specific application domains, such as ground, aerial, and underwater applications Michael et al. (2011); Smith et al. (2011); Girard et al. (2004); Grocholsky et al. (2006); Nigam and Kroo (2008). On work most closely related to ours, in Alamdari et al. (2012), iterative TSP paths are planned to minimize the maximum latency across all nodes in a connected network. The authors show that the approach yields $O(\log n)$ approximation on optimality in which n is the number of nodes in the network. The problem of generating speed profiles for robots along predetermined cyclic paths for keeping bounded the uncertainty of a varying field is addressed in Smith et al. (2012), in which the authors characterize appropriate policies for both single and multiple robots. In Soltero et al. (2012), decentralized adaptive controllers were designed to morph the initial closed paths of robots to focus on regions of high importance.

Sampling based planning methods (*e.g.* PRM, RRT, RRT* and their variations Kavraki et al. (1996); LaValle (1998); Karaman and Frazzoli (2011)) have also been applied to informative path planning problems. In Hollinger and Sukhatme (2014), Rapidly-Exploring Random Graphs (RRG) are combined with *branch and bound* methods for planning the most informative path for a mobile robot. In Lan and Schwager (2013), the authors tackle the problem of planning a cyclic trajectory for best estimation of a time-varying Gaussian Random Field, using a variation of RRT called Rapidly-Expanding Random Cycles (RRC).

Lastly, our problem, and OP in general, also has a *coverage* element. Coverage of a two-dimensional region has been extensively studied in robotics Choset (2000, 2001); Gabriely and Rimon (2003), as well as in purely geometric settings, for example, in Chin and Ntafos (1988), where the proposed algorithms compute the shortest closed routes for continuous coverage of polygonal interiors under an infinite visibility sensing model. Coverage with limited sensing range was also addressed later Hokayem et al. (2008); Ntafos (1991).

Contribution: This paper brings three main contributions:

- We introduce COP as a novel non-linear extension to OP, to model and harness time-invariant spatial correlations that are frequently present in informative path and policy planning problems. In particular, our formulation addresses the challenging problem of planning information maximizing tours for single and multiple robots under a limited travel budget.
- We provide complete mixed integer quadratic programming (MIQP) models for solving a QCOP, a quadratic instantiation of COP. These models, combined with a good off-the-shelf MIQP solver, yields an anytime algorithm that can effectively compute robot tour(s) over networks with tens to hundreds of nodes, for the optimal estimation of the underlying spatially corrected spatiotemporal fields.
- We demonstrate that our models and algorithms are effective over both simulated and empirical data sets.

In comparison to the conference publication Yu et al. (2014b), the current paper provides a much more thorough treatment of COP. From the perspective of the problem statement, we now give a cleaner and more general formulation. From the perspective of algorithmic solutions, we have developed an anytime algorithm and additional heuristics, which greatly boost the computational speed, allowing much larger problems to be solved. In the simulation study, a much more comprehensive evaluation of the algorithmic performance as well as simulations on real temperature data are now included.

Organization: The rest of the paper is organized as follows. In Section II, we formally introduce COP and QCOP. In Section III, we derive MIQP-based anytime algorithms for solving QCOP for single and multiple robots. In Sections IV, we perform extensive computational experiments to verify the correctness and evaluate the performance of our algorithmic solutions. We then illustrate how QCOP may be applied to solve realistic persistent monitoring tasks in Section V and conclude the paper in Section VI. Table I lists symbols that are frequently used in the paper.

II. PROBLEM STATEMENT

We study the problem of using mobile sensing robots to periodically survey spatially distributed locations (nodes), assuming that the quantities to be measured at the nodes come from a smooth spatiotemporal (scalar or vector) field. Due to spatial and temporal variations, such fields can be highly complex and dynamic. However, in applications involving large spatial domains (*e.g.*, terrains, road networks, forests, oceans, and so on), the underlying spatial domain often does not change. The observation allows us to work with the premise that nearby nodes have mostly *time-invariant* spatial correlations², even though the overall field may change significantly over time. Exploiting

²Here, we use the broad meaning of correlation, which could be, but is not necessarily, the correlation of random variables.

TABLE I
LIST OF FREQUENTLY USED SYMBOLS AND THEIR INTERPRETATIONS.

$V = \{v_i\}$	Node or point of interest, $ V = n$
$G = (V, E)$	Node network
\mathbf{p}_i	The two dimensional coordinate of v_i
r_i	Utility of knowing complete information about v_i
$\psi(v_i, t)$	Time-varying scalar field
$A = \{a_k\}$	Mobile robot k , $ \{a_k\} = m$
v_{b_k}	Start and end node for robot a_k
c_k	Travel budget for robot a_k
Π	π_1, \dots, π_m , a set of robot tours
$J(\Pi)$	The cost function for a given set of robot tours
w_{ij}	The weight measuring v_i 's influence on v_j
x_i	Binary variable indicating whether v_i is on a tour
x_{ij}	Binary variable indicating whether v_j is visited immediately after v_i
x_{ijk}	Binary variable indicating whether v_j is visited immediately after v_i , by robot a_k
u_i	Integer variable, $2 \leq u_i \leq n$, the order of v_i in a tour path, if used
u_{ik}	Integer variable, $2 \leq u_{ik} \leq n$, the order of v_i in a tour path, if used, by robot a_k
d_{ij}	Travel cost from v_i to v_j , maybe non-symmetric
α_{ij}, β_{ij}	Linear regression coefficients

these correlations, at any given time, it becomes possible to infer the field's value at a certain node from the values of adjacent nodes.

Before formally stating the problem, roughly speaking, we are interested in deploying mobile sensors with limited travel range to sample nodes of a network. Based on the samples at the nodes, we then infer the field's value at the rest of the nodes from correlation when possible. Besides time-invariant spatial correlations, we assume that the field remains relatively static during a single trip of the mobile sensor(s). We denote this problem the *Correlated Orienteering Problem* (COP). After introducing the broad COP problem, we focus on a special type of COP with a quadratic cost function induced by independent, linear correlations between adjacent nodes. We call this special instantiation of COP the *Quadratic Correlated Orienteering Problem* (QCOP). Below, COP and QCOP are formally defined.

A. Correlated Orienteering Problem

Let $V = \{v_1, \dots, v_n\}$ be a set of spatially distributed nodes in some workspace $W \subset \mathbb{R}^2$. Each node $v_i \in V$ is associated with coordinates $\mathbf{p}_i \in \mathbb{R}^2$. Let $\psi(\mathbf{p}, t), \mathbf{p} \in \mathbb{R}^2$ be a time varying scalar field over W . The values on the nodes of V , with a slight abuse of notation, are written as $\psi(v_i, t), 1 \leq i \leq n$. We assume that the spatial relationship among the nodes of V , as determined by ψ , induces a directed graph G over V . More precisely, $G = (V, E)$ has an edge (v_i, v_j) if and only if $\psi(v_j, t)$ is dependent on $\psi(v_i, t)$. That is, let $N_i = \{v_{i_1}, \dots, v_{i_k}\}$ be the set of neighbors of v_i in G , with edges pointing to v_i , then for some time-invariant f_i ,

$$\psi(v_i, t) = f_i(\psi(v_{i_1}, t), \dots, \psi(v_{i_k}, t)). \quad (1)$$

Let there be m mobile robots, $A = \{a_1, \dots, a_m\}$. Each robot follows the standard single integrator dynamics with constant magnitude on the control input (*i.e.*, $\dot{x} = u$ with $\|u\| = 1$). To model the travel distance constraints inherent with mobile robots, for a robot a_k , let its base (*i.e.*, where it must start and end in a cyclic tour) be a node $v_{b_k} \in V$. Let

$$c : A \rightarrow \mathbb{R}^+, a_k \mapsto c_k$$

represent the maximum distance (budget) the mobile robots can travel before they must return to their respective bases. Other than the single integrator dynamics and the distance budget constraints, these robots have no other motion constraints. In particular, a robot is not constrained to the implicitly defined graph G and can travel in a straight line between any $\mathbf{p}_i, \mathbf{p}_j \in W$ as permitted by the travel distance budget.

A mobile robot can measure $\psi(v_i, t)$ when the robot is located at $v_i \in V$. Let $\Pi = \{\pi_1, \dots, \pi_m\}$ be a set of tours in which each π_k is a cyclic path for a_k that goes through a set of nodes of V including v_{b_k} . The overall quality of Π is measured by some utility function $J: \{\Pi\} \rightarrow \mathbb{R}^+ \cup \{0\}$ that maps path sets to non-negative real values. We do not consider sensor measurement noises in the current paper.

The *Correlated Orienteering Problem* (COP) is defined as a 4-tuple $(V, \psi, \{v_{b_k}\}, J)$ over which we wish to find a set of tours Π that maximizes the utility $J(\Pi)$. Note that ψ is fixed but generally unknown; it can only be measured (by mobile robots) at particular locations and time instances. An illustrative and qualitative example of what COP aims to achieve is given in Figure 2.

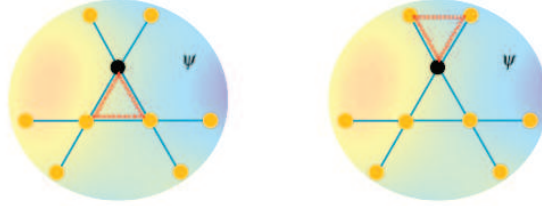


Fig. 2. Two tours with the same budget over the same node network and ψ . Here, all edges have unit length. Given a budget of 3 with the black node as the starting node, intuitively, with correlations of node values between nearby nodes, if we want to estimate ψ over all 9 nodes, the tour (dashed path) on the left is likely better because each of the 9 nodes is adjacent to some node visited by the tour. COP aims to allow the planning of such a tour Π through the maximization of a properly defined $J(\Pi)$.

Remark. At a first glance, COP may appear to mimic a problem whose underlying process is a Markov Decision Process (MDP). Although there are some similarities between the two formulations (*e.g.*, like in an MDP-based problem, in COP, the robots take actions to go to different physical states and the information to be collected follows some distribution), a key difference is that it is never beneficial to revisit a node in a COP instance but revisiting a state in an MDP problem can be rewarding. This is also a key source of computational difficulty associated with COP because dynamic programming techniques that are useful in solving MDP problems can no longer be applied to COP.

B. Quadratic Correlated Orienteering Problem

After defining the general COP problem class, we now describe an instantiation of COP with a quadratic utility (alternatively, reward or cost) function. If the tours of the m robots, Π , go through a node $v_i \in V$, then a utility of $r(v_i)$ is collected, defined according to the mapping

$$r: V \rightarrow \mathbb{R}^+, v_i \mapsto r_i.$$

The robots do not gain more utility for revisiting v_i in Π . To represent the total utility of QCOP, let $\{x_1, \dots, x_n\}$ be n binary variables, with $x_i = 1$ if and only if v_i (*i.e.*, \mathbf{p}_i) is on some tour $\pi_k \in \Pi$. To incorporate correlations among the nodes during the tour planning phase while also rendering the formulation more concrete, we let ψ (and therefore $\{f_i\}$) and J have the following instantiation. Let the weight function

$$w: E \rightarrow \mathbb{R}^+, (v_j, v_i) \mapsto w_{ji}$$

represent the effectiveness of using $\psi(v_j, t)$ to estimate $\psi(v_i, t)$. One may view w_{ji} as representing the amount of information that $\psi(v_j, t)$ has about $\psi(v_i, t)$, independent of other neighbors of v_i . The utility that can be collected over a node v_i is defined as

$$r_i \left(x_i + \sum_{v_j \in N_i} w_{ji} x_j (x_j - x_i) \right), \quad (2)$$

in which the quadratic term $x_j(x_j - x_i)$ is non-zero if and only if $x_j = 1$ and $x_i = 0$. Essentially, (2) says that correlations is only relevant for v_i if v_i is not directly visited by a robot. Obviously, for each $0 \leq i \leq n$, $\sum_{v_j \in N_i} w_{ji} \leq 1$.

Note that we do not assume that $w_{ij} = w_{ji}$, which may be the case if the field's values at v_i and v_j are, for example, jointly Gaussian random variables. Over a set of tours for the robots, Π , the total utility to be maximized is then

$$J(\Pi) = \sum_{i=1}^n \left(r_i x_i + \sum_{v_j \in \mathcal{N}_i} r_j w_{ij} x_i (x_i - x_j) \right). \quad (3)$$

We observe that the utility function (3) defines a natural quadratic extension to OP, which has a linear utility $\sum_{i=1}^n r_i x_i$. We denote this COP instantiation as the *Quadratic Correlated Orienteering Problem* (QCOP). By assuming independence among w_{ij} 's in formulating QCOP, we trade model fidelity for computational efficiency. Nevertheless, we note that QCOP is still a difficult problem computationally.

Theorem 1 *COP and QCOP are NP-hard.*

PROOF. It is straightforward to observe that OP, for even a single robot (player), is NP-hard Vansteenwegen et al. (2011). To see this, for a given travel budget, an algorithm solving OP for a single robot must implicitly answer the question of whether the budget is enough for the robot to go through all nodes in V . Therefore, OP contains as a sub-problem the Euclidean traveling salesman problem (TSP), which is NP-hard.

For QCOP with the quadratic utility given by Equation (3), making the weights $\{w_{ij}\}$ sufficiently small reduces it to an OP, because the quadratic utility (the second summation in Equation (3)) then becomes negligible. This shows that a general COP is also NP-hard. \square

III. MIXED INTEGER QUADRATIC PROGRAMMING MODELS FOR QUADRATIC COP

In this section, we propose a quadratic integer programming model with quadratic utility functions and linear constraints (often known as *mixed integer quadratic programming* or MIQP) for solving QCOP with the utility function given by Equation (3). We start from the case of a single mobile robot and then move to the case of multiple robots. Then, we discuss the associated algorithmic aspects and provide an algorithm outline.

A. MIQP Model for a Single Robot

We start with $m = 1$ (single robot) and adapt the constraints from Vansteenwegen et al. (2011), which yields a compact model. Whereas our description of the model is self-contained for completeness, more background knowledge on the development of linear OP models can be found in Vansteenwegen et al. (2011) and the references within.

Without loss of generality, let the robot start from v_1 . Let x_{ij} be a binary variable with $x_{ij} = 1$ if and only if the robot visits v_j immediately after it visits v_i . Recall that this does not depend on the existence of an edge between v_i and v_j in G . Because it is never beneficial to revisit a node, the robot must only enter and leave any node at most once. This allows us to represent the number of times that the robot enters (resp. leaves) a node i as $\sum_{j=1, j \neq i}^n x_{ij}$ (resp. $\sum_{j=1, j \neq i}^n x_{ji}$). Both of these quantities can be at most one. The tour constraint then says that these two quantities must be equal, *i.e.*, $\sum_{j=1, j \neq i}^n x_{ij} = \sum_{j=1, j \neq i}^n x_{ji}$. Recall that x_i is the binary variable indicating whether v_i is visited, we obtain the following constraints

$$\sum_{j=1, j \neq i}^n x_{ij} = \sum_{j=1, j \neq i}^n x_{ji} = x_i \leq 1, \quad \forall 2 \leq i \leq n. \quad (4)$$

For $i = 1$, because v_1 must be visited, we have

$$\sum_{i=2}^n x_{1i} = \sum_{i=2}^n x_{i1} = x_1 = 1. \quad (5)$$

The constraints (4) and (5) ensure that the robot will take a tour starting and ending at v_1 . They do not, however, prevent multiple disjoint tours from being created. To prevent this from happening, let $2 \leq u_i \leq n$ be *integer* variables for $2 \leq i \leq n$. If there is a single tour starting from v_1 , then u_i can be chosen to satisfy the constraints

$$u_i - u_j + 1 \leq (n-1)(1 - x_{ij}), \quad 2 \leq i, j \leq n, i \neq j \quad (6)$$

To see that this is true, note that since $u_i - u_j + 1 \leq n - 1$ regardless of the values taken by $2 \leq u_i, u_j \leq n$, (6) can only be violated if $x_{ij} = 1$. The condition $x_{ij} = 1$ only holds if v_j is visited immediately after v_i is visited. Setting u_i to be the order with which v_i is visited on the tour, if $x_{ij} = 1$, then $u_i - u_j + 1 = 0$, satisfying (6). On the other hand, if there is another tour besides the one starting from v_1 and when $v_{ij} = 1$, then the RHS of (6) equals zero. For (6) to hold, we must have $u_i - u_j + 1 \leq 0 \Rightarrow u_i < u_j$. However, this condition cannot hold for all consecutive pairs of nodes on a cycle. Thus, (6) enforces that only a single cycle may exist.

With the introduction of the variables $\{x_{ij}\}$, the tour distance constraint can be enforced via

$$\sum_{i=1}^n \sum_{j=1, j \neq i}^n x_{ij} d_{ij} \leq c_1, \quad (7)$$

in which d_{ij} is the distance from v_i to v_j and c_1 is the tour distance constraint for the single (first) robot. Note that the distance d_{ij} needs not to be symmetric. Moreover, it is straightforward to incorporate sensing cost at a node v_i by absorbing that cost into d_{ij} for all $j \neq i$. Alternatively, if the sensing cost is not compatible with the travel cost, an additional cost constraint can be added as well. Putting things together, we obtain a complete MIQP model for QCOP, summarized below.

$$\begin{aligned} & \text{maximize } J(\Pi) = \sum_{i=1}^n \left(r_i x_i + \sum_{v_j \in N_i} r_j w_{ij} x_i (x_i - x_j) \right) \\ & \text{subject to} \\ & \quad \sum_{j=1, j \neq i}^n x_{ij} = \sum_{j=1, j \neq i}^n x_{ji} = x_i \leq 1, \quad \forall 2 \leq i \leq n \\ & \quad \sum_{i=2}^n x_{1i} = \sum_{i=2}^n x_{i1} = x_1 = 1 \\ & \quad u_i - u_j + 1 \leq (n-1)(1 - x_{ij}), \quad 2 \leq i, j \leq n, i \neq j \\ & \quad \sum_{i=1}^n \sum_{j=1, j \neq i}^n x_{ij} d_{ij} \leq c_1 \end{aligned} \quad (8)$$

B. MIQP Model for Multiple Robots

Extending a single tour to multiple tours is rather straightforward. To accommodate m robots, the variables $\{x_{ij}\}$ and $\{u_i\}$ are extended to x_{ijk} and u_{ik} , with $1 \leq k \leq m$ representing the robots. Constraints (4) and (5) become

$$\sum_{j=1, j \neq i}^n x_{ijk} = \sum_{j=1, j \neq i}^n x_{jik} \leq 1, \quad \forall 1 \leq i \leq n, 1 \leq k \leq m, \quad (9)$$

$$\sum_{k=1}^m \sum_{j=1, j \neq i}^n x_{ijk} = \sum_{k=1}^m \sum_{j=1, j \neq i}^n x_{jik} = x_i \leq 1, \quad \forall 1 \leq i \leq n, \quad (10)$$

and

$$\sum_{i=1, i \neq b_k}^n x_{b_k i k} = \sum_{i=1, i \neq b_k}^n x_{i b_k k} = x_{b_k} = 1, \quad 1 \leq k \leq m. \quad (11)$$

Equation (4) splits into Equations (9) and (10) because we need Equations (9) to ensure that a node is used by at most one robot. With only (10) but not (9), it can happen that one robot enters a node while a different robot exits the same node, which should not happen.

The constraints on u_{ik} become (for all $1 \leq k \leq m$)

$$u_{ik} - u_{jk} + 1 \leq (n-1)(1 - x_{ijk}), \quad i, j \neq b_k, i \neq j, 1 \leq i, j \leq n. \quad (12)$$

The traveled distance constraint, Equation (7), becomes

$$\sum_{i=1}^n \sum_{j=1, j \neq i}^n x_{ijk} d_{ij} \leq c_k, \quad 1 \leq k \leq m. \quad (13)$$

Finally, the utility function Equation (3) remains the same.

Remark. The above MIQP model for QCOP does not allow two robots to start from the same base. We can easily accommodate such scenarios via modifying Equations (9), (10), and (11) accordingly.

C. Algorithmic Perspectives

In this subsection, we discuss several important algorithmic aspects that were explored in the implementation of MIQP models, aiming at improving the performance of the algorithm. An algorithm outline is also provided.

Anytime Property: An interesting and extremely useful property, which also comes naturally from our MIQP-based method, is that it leads to an *anytime* algorithm. Integer linear and quadratic programming solvers, which are often variations of branch-and-bound algorithms, work with a polytope containing all feasible solutions to the (relaxed) continuous optimization problem. The algorithm functions by braking the polytope into smaller pieces and evaluate the objective function on each of these pieces. As the algorithm progresses, more and more of the initial polytope gets truncated. After some initial steps, a tree structure is built and the leaves of the tree contain active portions of the original feasibility polytope. For each of these pieces (which are again polytopes), in a maximization problem, it is relatively easy to locate a feasible solution with the correct integrality condition (*i.e.*, a feasible solution in which binary/integer variables get assigned binary/integer values). The maximum objective value from all these feasible solutions is then a lower bound of the optimal value. At the same time, without respecting the integrality constraints, the maximum achievable objective can also be computed for each leaf polytope. This yields a lower bound on the optimal value. The difference between the two bounds is often referred to as the *gap*. An optimal solution is found when this gap reaches zero. If the gap gradually decreases in the execution of a branch-and-bound algorithm, which is the case in our problem, an anytime algorithm is obtained.

For a difficult optimization problem like QCOP, having an anytime algorithm with known optimality gap is beneficial for at least two reasons. First, because it generally takes more and more time for an algorithm to find better and better solutions, having the option to stop early at a sub-optimal solution can save a significant amount of time. This is the *anytime* perspective. Second, because we know exactly how optimal our solution is at an arbitrary time instance during the execution process, we may stop running the algorithm and have the confidence that a desired level of optimality is achieved. Note that anytime algorithms are not always equipped with quantitative characterizations of how optimal their current solutions are. Our anytime algorithm, on the other hand, always maintains a good estimation on how optimal the current best solution (also known as the incumbent) is.

Algorithm Outline: Putting things together, we obtain the tour planning and node value estimation algorithm outlined in Algorithm 1. In lines 1- 2, the MIQP model is set up and solved to obtain the desired set of tours for the robots. The robots then follow the planned tours and collect data as they pass over the nodes on these tours in line 3. The collected data $\{\psi'(v_i, t_s)\}$ is subsequently updated through correlation to yield the final estimated node values. Note that `UpdateNodeEstimate(\cdot)` is only determined when QCOP is connected to a particular spatiotemporal field $\psi(\cdot, \cdot)$ (an example is given in Section V).

Algorithm 1: QCOP ESTIMATION

Input : $G = (V, E)$: node network, $|V| = n$
 $W = \{w_{ij}\}, 1 \leq i, j \leq n$: correlation weights
 $V_B = \{v_{b_1}, \dots, v_{b_m}\}$: base nodes
 $C = \{c_1, \dots, c_m\}$: travel distance budgets
 gap : optimality tolerance

Output: $\psi'(v_i, t_s), 1 \leq i \leq n$: node estimation at time t_s

%Compute robot tours

1 $M \leftarrow \text{SetUpModel}(G, W, V_B, C)$; %Set up model
2 $\{\pi_1, \dots, \pi_{b_m}\} \leftarrow \text{SolveModel}(M, gap)$; %Compute tours

%Run tours and collect data

3 $\{\psi'(v_i, t_s)\} \leftarrow \text{CollectData}(\{\pi_1, \dots, \pi_{b_m}\})$

%Estimate value at unvisited nodes

4 $\{\psi'(v_i, t_s)\} \leftarrow \text{UpdateNodeEstimate}(\{\psi'(v_i, t_s)\})$

5 **return** $\{\psi'(v_i, t_s)\}$

Encoding Utility in Linear Form: We observe that the utility given by (3) can in fact be turned into a linear utility without loss of accuracy. To do so, for each (i, j) pair, define an additional binary variable z_{ij} . We may then enforce $z_{ij} = 1$ if and only if $x_i(x_i - x_j) = 1$ by adding two constraints,

$$z_{ij} \leq x_i, \quad (14)$$

$$z_{ij} \leq \frac{x_i - x_j + 1}{2}, \quad (15)$$

to our model. Clearly, if $x_i = 0$ or $x_j = 1$, these constraints ensure that $z_{ij} = 0$. On the other hand, updating (3) to

$$J(\Pi) = \sum_{i=1}^n \left(r_i x_i + \sum_{v_j \in N_i} r_j w_{ij} z_{ij} \right), \quad (16)$$

when $x_i = 1$ and $x_j = 0$, maximizing $J(\Pi)$ ensures that $z_{ij} = 1$.

Having a linear utility effectively transforms our MIQP model into a mixed integer linear programming (MILP) model. Whereas the transformation does not directly reduce problem complexity, it can be beneficial for the solver to be aware that the problem is in fact an MILP.

Restricting the Travel Distance Between Nodes: During the computational evaluation of our algorithm, we observe that, when utilities of the nodes are relatively similar, an optimal robot path rarely moves between two nodes that are too far away. In fact, due to the local correlation model, when an optimal path moves from one node to another, the second node is almost always within the 2-neighborhood³ of the first node. Based on this observation, we attempted a heuristic that restricts how far a robot may travel from one node to the next. With the heuristic, each node then creates a constant number x_{ij} variables in the MIQP model. Thus, the size of the MIQP model is greatly reduced with the introduction of this heuristic.

IV. MODEL CORRECTNESS AND PERFORMANCE

In this section, through computational experiments, we further validate the correctness of our MIQP models for QCOP and evaluate computational performance of these models. Note that in this section our focus is on $J(\Pi)$ and we do not actually collect data and perform node value estimation (*i.e.*, effectively, we only run lines 1-2 of Algorithm 1). Unless otherwise stated, the quadratic utility (3) is used instead of the linear utility (16). We implemented the linear and quadratic models in the JAVA programming language interfacing with the Gurobi solver Gurobi Optimization Inc. (2014). All computations were performed on a Dual-Intel Xeon E5-2623 workstation under an 8GB JavaVM.

A. Model Correctness

Regular Grids: We first briefly show that our MIQP model-based algorithms indeed maximize the objective function given by (3). We begin with a single robot and use 3×3 and 4×4 grid networks with unit edge length as the test node networks. We set the weights w_{ji} for a node i simply as $1/|N_i|$. For example, if node i has three neighbors, then all w_{ji} 's are set to $1/3$. For the 3×3 grid, we let the single robot start at the middle node on

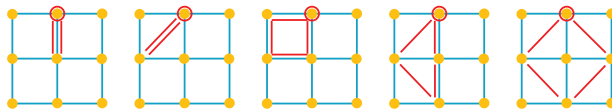


Fig. 3. Single robot tours for travel budgets 2, 3, 4, 5, and 6, in that order, from left to right.

the top row (the circled node in Figure 3) and let the maximum allowed travel budget vary from 2 to 6 with unit increments. Each node has a unit utility. The computed tours for these budgets are illustrated in Figure 3, with utilities as 4.0, 4.5, 5.7, 7.3, and 9 (maximum possible), respectively. One can easily verify that these are consistent with the design of the single-robot MIQP model. For the 4×4 grid, under a similar setup, we get the tours as illustrated in Figure 4 for travel budgets 4, 8, and 12, respectively. The associated maximum utilities are 6.2, 11.5, and 16.0, respectively.

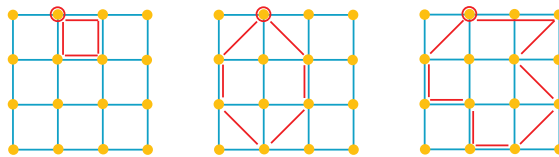


Fig. 4. Single robot tours for travel budgets 4, 8, and 12, in that order, from left to right.

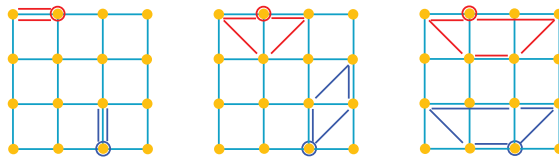


Fig. 5. Two-robot tours for travel budgets (per robot) 3, 5, and 7, in that order, from left to right.

Next, a two-robot setup is tested on the 4×4 grid, with the robots starting at opposite locations as indicated by the red and purple circled nodes in Figure 5, which illustrates the tours with individual travel budgets 3, 5, and 7, respectively. The associated maximum utilities are 7.2, 12.5, and 16.0, respectively. Each problem instance in this subsection took at most two seconds to solve.

Irregular Node Network: Our second experiment works with the irregular node network from Figure 1. The bounding rectangle of the network is roughly 13 units by 8 units. For this network, weights (w_{ij} 's) are again computed based on the number of neighbors. Up to three robots were attempted with the longest running time being about 100 seconds. The trial results and the associated parameters are given in Figure 6. The base nodes, indicated as colored circles, were hand picked (only once, *i.e.*, we did not try any other choices and then select the best one) to be roughly evenly distributed on the network.

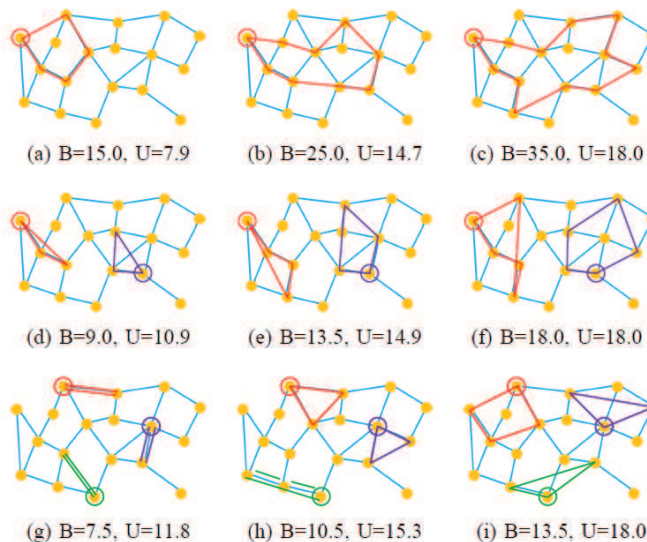


Fig. 6. Results from running MIQP models for QCOP on the irregular, realistic node network from Figure 1. The numbers under each picture indicate the budget (B) per tour/robot and the total utility (U), respectively.

From the result (Figure 6), we see that the MIQP model always selects tours that do not have spatial overlaps, which is expected but a nice feature to have nevertheless. Also, regardless of the number of robots and tours, the total travel budgets (35.0 for one tour, 36.0 for two tours, and 40.5 for three) to ensure full coverage of the network appear to be similar. We note that for the cases with multiple tours, some of the individual budgets can be

³A node v_j is in the *2-neighborhood* of a node v_i if there exists $v_k \in N_i$ (N_i is the set of immediate neighbors of v_i) such that $v_j \in N_j$.

shortened. For example, the tours in Figure 6(f) can be (manually) updated to the tours in Figure 7, with reduced total (actual) travel budget but without reducing the collected utility. Varying individual budgets is supported by our model by default.

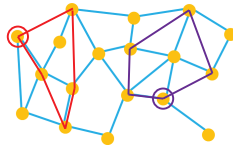


Fig. 7. Updated two-robot tours with the same utility as the tours in Figure 6(f).

B. Computational Performance, Regular Grid

Computing Exact (Optimal) Solutions: We now look at the performance of our MIQP/MIP models for solving QCOP, starting with computing the exact solution. For all computations, we set a 2,500-second time limit. To evaluate the computational performance, we again start with a single robot and attempt grid networks with different sizes. For networks with up to 6×6 nodes, our MIQP model can compute optimal solutions for all choices of budgets within 10 minutes. We note that a 36-node network is applicable to many realistic scenarios as demonstrated in Section V. The performance characteristics are listed in Table II.

TABLE II
PERFORMANCE, REGULAR GRIDS AND A SINGLE ROBOT

Grid		Trial #			
		1	2	3	4
4×4	budget/utility	3.2/4.3	6.4/9.7	9.6/13.7	12.8/16.0
	time(s)	0.06	0.40	0.72	0.03
5×5	budget/utility	8.0/12.1	12.0/18.0	16.0/23.3	20.0/25.0
	time(s)	2.4	8.9	5.1	0.3
6×6	budget/utility	9.6/14.2	14.4/22.3	19.2/28.3	24.0/34.0
	time(s)	63.6	45.3	243	217
7×7	budget/utility	11.2/16.5	16.8/-		
	time(s)	550	>2,500		

Because the MIQP model requires one extra set of variables for each extra robot, computing exact solutions for multiple robots becomes more challenging as the number of robots increases. Table III lists the complete computational results on the same grids for two robots and up to 5×5 grids with various budgets. It becomes fairly infeasible to compute for grids of size 6×6 and beyond with two robots. For three robots (we omit the limited detail here), it becomes challenging to compute exact solutions for all budgets over a 5×5 grid.

TABLE III
PERFORMANCE, REGULAR GRIDS AND TWO ROBOTS

Grid		Trial #			
		1	2	3	4
4×4	budget/utility	3.2/7.2	4.8/10.9	6.4/14.6	8.0/16.0
	time(s)	0.08	0.17	5.3	0.06
5×5	budget/utility	4.0/11.1	6.0/15.8	8.0/20.5	10.0/24.3
	time(s)	0.11	34.8	1070	1961

To get a better grasp at the overall performance of the exact algorithm, we randomly perturb regular grids to obtain node networks such as the one shown in Figure 8. This allows us to get random test cases that mimic more

realistic scenarios. Since a regular 7×7 grid proves to be time-consuming to work with, we focus on network with up to 36 nodes. The result is listed in Table IV. For each grid size and budget combination, the utility and the computation time are averages over ten randomly generated instances. Comparing Table II with Table IV, we observe that instances with randomized grids and regular grids seem to have similar computational complexity.

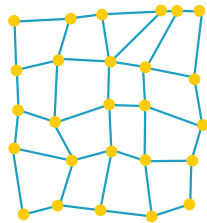


Fig. 8. A perturbed 5×5 grid network.

TABLE IV
PERFORMANCE, RANDOM GRIDS AND A SINGLE ROBOT

Grid		Trial #			
		1	2	3	4
4×4	budget/utility	3.6/5.1	7.1/10.7	10.7/15.4	14.2/16
	time(s)	0.05	0.90	0.41	0.14
5×5	budget/utility	4.4/6.3	8.9/13.5	13.3/20.3	17.8/25.0
	time(s)	0.19	9.9	16.0	3.6
6×6	budget/utility	10.7/16.4	16/25.6	21.3/32.7	26.7/36
	time(s)	195	177	83.7	3.7

Performance of Anytime Algorithms: Once we are willing to allow a small amount of deviation from the true optimal solution, which is perfectly acceptable for practical purposes, the anytime property can greatly boost the computational performance. In our tests, we set the optimality gap to be 0.2, meaning that the solution is at most 20% worse than the optimal solution (note that we may still stop the algorithm at any time and have an intermediate solution). Our first experiment compares the anytime algorithm with the exact algorithm by executing the program over the same set of instances used for obtaining the result listed in Table IV. The performance of the anytime algorithm is listed in Table V. In comparison to Table IV, the loss of optimality is often much less than the set 20% threshold. At the same time, the computation time is improved up to 30 times. The improvement become more obvious as the grid size grows.

TABLE V
PERFORMANCE, ANYTIME, RANDOM GRIDS AND A SINGLE ROBOT

Grid		Trial #			
		1	2	3	4
4×4	budget/utility	3.6/5.1	7.1/10.6	10.7/14.1	14.2/14.9
	time(s)	0.01	0.52	0.22	0.07
5×5	budget/utility	4.4/6.2	8.9/13.2	13.3/19.2	17.8/21.7
	time(s)	0.15	5.2	3.4	0.7
6×6	budget/utility	10.7/16.0	16/24.3	21.3/31.6	26.7/31.2
	time(s)	43.7	13.3	2.9	1.7

The speed improvement only widens as the problems get bigger. In Table VI, we test the anytime algorithm on regular-grid instances (the rest of our tests in this section work only with regular grids, since they appear to be of similar difficulty as instances over random grids) with up to about 150 nodes. All instances were solved with the

longest running time being about 12 minutes. The result suggests that if we settle for near-optimal solutions, very large problem instances can be solved quickly.

TABLE VI
PERFORMANCE, ANYTIME, REGULAR GRIDS AND A SINGLE ROBOT

Grid		Trial #			
		1	2	3	4
6×6	budget/utility	10.7/15.5	16.0/22.9	21.3/30.2	26.7/32.5
	time(s)	16.3	5.0	3.8	1.5
8×8	budget/utility	14.2/19.4	21.3/31.3	28.4/39.7	35.6/50.6
	time(s)	112	169	162	17.3
10×10	budget/utility	17.8/26.3	26.7/39.6	35.6/49.3	44.4/64.2
	time(s)	217	546	236	270
12×12	budget/utility	21.3/31.7	32.0/48.5	42.7/61.7	53.3/79.5
	time(s)	722	235	669	300

With the anytime algorithm, more challenging multi-robot problems can be solved. For two robots, with 20% optimality tolerance, regular grids with sizes up to 7×7 can now be tackled. The performance is similar for three robots. The results are listed in Tables VII and VIII.

TABLE VII
PERFORMANCE, ANYTIME, REGULAR GRIDS AND TWO ROBOTS

Grid		Trial #			
		1	2	3	4
5×5	budget/utility	4.0/10.3	6.0/15.8	8.0/19.8	10.0/21.1
	time(s)	0.06	13.2	9.6	3.0
6×6	budget/utility	4.8/10.9	7.2/19.3	9.6/25.3	12.0/30.0
	time(s)	0.16	225	322	18.1
7×7	budget/utility	5.6/14.6	8.4/23.0	11.2/30.9	14.0/37.6
	time(s)	144	1329	1640	871

TABLE VIII
PERFORMANCE, ANYTIME, REGULAR GRIDS AND THREE ROBOTS

Grid		Trial #			
		1	2	3	4
5×5	budget/utility	2.7/10.3	4.0/15.3	5.3/19.3	6.7/21.7
	time(s)	0.05	0.11	1.1	0.89
6×6	budget/utility	3.2/11.3	4.8/17.0	6.4/25.8	8.0/30.3
	time(s)	0.08	0.44	82.8	26.5
7×7	budget/utility	3.7/14.2	5.6/22.4	7.5/31.8	9.4/-
	time(s)	0.2	514	514	>2,500

Linear Utility Function and Restricted Travel Distance: With everything being equal, having a linear model instead of a quadratic one generally boosts computational performance because solvers for linear models are more efficient in general. On the other hand, in our particular case, turning the MIQP model to a MILP model introduces a significant number (usually about one-third more) of additional variables, which unfortunately increases the size of the model quite a lot. In practice, we noticed that linear utility by itself does not improve the computational performance by much, unless the problem instance is large (*i.e.* > 50 nodes). Even though a several-fold speed

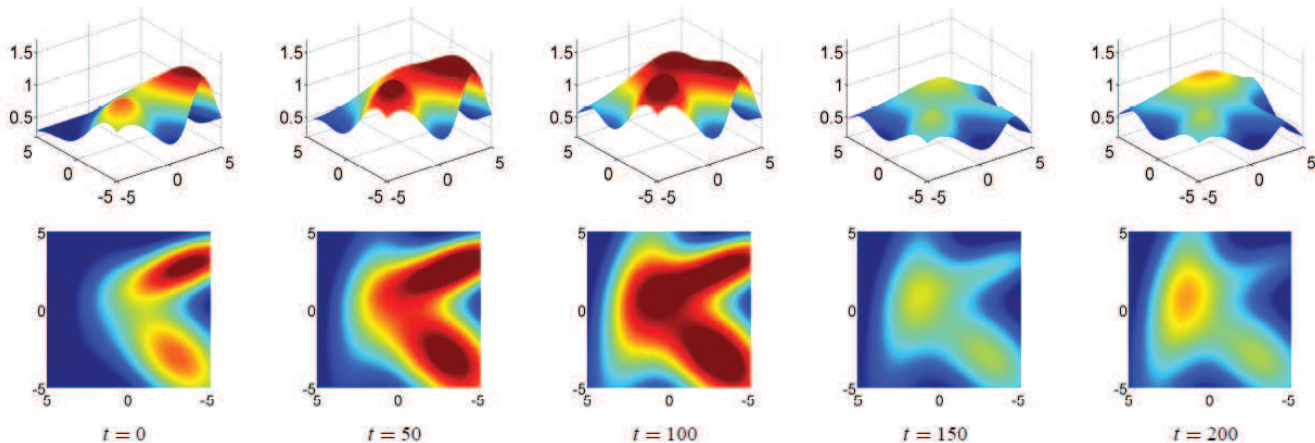


Fig. 9. Snapshots of a synthetic scalar field at time steps 0, 50, 100, 150, and 200, from left to right, respectively. [top row] Three-dimensional views. [bottom row] Two-dimensional heat-map views.

boost was observed (we omit the limited result here), these instances still take hours to compute using a MILP formulation.

We also experimented heavily on restricting the travel distance between nodes. Somewhat surprisingly, the reduction of model size from this heuristic, usually over several folds, does not substantially alter the running time. Interestingly, however, when we combine these two heuristics with the anytime algorithm, something of practical importance came out. For instances with up to 100 nodes, significant performance increase is observed. The result is listed in Table IX. When compared with Table VI, we observe that the computation time is greatly reduced without noticeable impact on the solution quality.

TABLE IX
PERFORMANCE, ANYTIME WITH HEURISTICS, REGULAR GRIDS AND A SINGLE ROBOT

Grid		Trial #			
		1	2	3	4
6 × 6	budget/utility	10.7/15.3	16.0/23.8	21.3/30.0	26.7/30.8
	time(s)	7.5	1.8	0.9	0.1
8 × 8	budget/utility	14.2/20.7	21.3/31.2	28.4/41.3	35.6/51.5
	time(s)	13.8	10.2	18.2	11.1
10 × 10	budget/utility	17.8/26.3	26.7/39.7	35.6/52.4	44.4/64.1
	time(s)	24.8	13.1	89.6	167

V. APPLICATIONS TOWARD PERSISTENT MONITORING

In this section, we demonstrate how QCOP and our algorithm may be applied to realistic persistent monitoring problems. Because our focus here is on estimation quality, we apply the the exact algorithm on a single mobile robot. To be extensive, we include one simulation experiment performed over a synthetic spatiotemporal field and one simulation experiment using real temperature data from 14 weather stations in the state of Massachusetts. When it comes to applying QCOP to persistent monitoring tasks, we must first obtain $\{w_{ij}\}$ from historical data collected over all nodes, which is a *learning* problem. Recall that $N_i := \{v_j \mid (v_i, v_j) \in E\}$ is the neighbor set of $v_i \in V$. We apply a linear regression model over ψ , *i.e.*,

$$\psi(v_i, t) = \alpha_{0i} + \sum_{v_j \in N_i} \alpha_{ji} \psi(v_j, t), \quad (17)$$

in which α_{0i} and α_{ji} 's are coefficients to be computed from historical data with T sets of node value data. (17) corresponds to models such as the Gaussian Process (GP). To map these coefficients to QCOP, for each w_{ji} , we

compute two sets of such coefficients. The first set of coefficients α'_{ji} 's are computed assuming all of N_i are visited; the second set, α''_{ji} 's, are computed assuming v_j is v_i 's only visited neighbor. We then compute the weight w_{ji} via

$$w_{ji} = \frac{\alpha'_{ji} + \alpha''_{ji}}{\sum_{k \in N_i} (\alpha'_{ki} + \alpha''_{ki})}. \quad (18)$$

Equation (18) was chosen to balance the impact of single neighbors as well as the impact of the entire neighborhood.

With $\{w_{ij}\}$, for a given travel budget, a utility maximizing tour can be computed. Assuming the robot collects exact values at time t_s from nodes it visits, the values on nodes that are not visited by the robot are estimated as follows. Let v_i be such a node and let N'_i be its neighbor set such that a node $v_j \in N'_i$ has either measured or estimated value (at time t_s). The historical data for v_i and nodes in N'_i from $1 \leq t \leq T$ are then used to perform multiple linear regression according to

$$\psi(v_i, t) = \beta_{0i} + \sum_{v_j \in N'_i} \beta_{ji} \psi(v_j, t). \quad (19)$$

The obtained β_{0i} and β_{ji} 's can then be used to compute the estimated $\psi'(v_i, t_s)$ using (19). The process is repeated until all nodes are covered. This iterative process defines the function `UpdateNodeEstimate(.)` in Algorithm 1.

A. Measuring a Time-Varying Scalar Field

Our first application-driven simulation verifies the effectiveness of Equation (18) in connecting actual scalar fields to QCOP. Our experiments are performed over a synthetic scalar field generated by three two-dimensional Gaussians. These Gaussians have fixed centers but varying intensities and covariance matrices over time; we fix the centers to ensure that the spatial correlations are relatively time-invariant. The field is simulated for 201 time steps; the snapshots of the field at time steps 0, 50, 100, 150, and 200 are provided in Figure 9. The node network used here is a 5×5 randomized grids (see, e.g., Figure 8) scaled to the dimensions of the support of the scalar field. For each fixed travel budget, 100 random 5×5 node networks are generated. In each randomly generated network, the nodes of the network are given equal importance (i.e., unit utility). To estimate α'_{ij} 's and α''_{ij} 's for computing the weights, data from the first fifty time steps were used ($T = 50$). For running the model to obtain a utility maximizing tour, the second diagonal node from the top-left corner was used as the base node. The resulting tour is then used to obtain $\psi'(v_i, t_s)$ for $t_s = 100, 150, \text{ and } 200$, according to (19). We define the *quality* of $\psi'(v_i, t_s)$ as

$$\frac{\sum_{t_s \in \{100, 150, 200\}} (\psi(v_i, t_s) - |\psi'(v_i, t_s) - \psi(v_i, t_s)|)}{\sum_{t_s \in \{100, 150, 200\}} \psi(v_i, t_s)}. \quad (20)$$

To compare to our results, we also exhaustively search through the network for a tour starting and ending at the same base node that minimizes the same quality defined by Equation (20) under the same travel budget. This experiment was limited to travel budgets 6 and 8, corresponding to tours containing up to five nodes. While our model can produce tours with many more nodes, for comparing the result, we have to exhaustively search through all tours starting from the base node to find the best one, which becomes very costly as the number of nodes is over 5. The quality score obtained this way is denoted as ‘‘actual quality’’. The result comparing the approaches is given in Table X. Using the given metric, the average quality error was less than one, meaning that it was not more than the error incurred by omitting a single node. In roughly 30% of the cases, the tour found using our method was identical to the one found using exhaustive tour search.

TABLE X
MODEL FIDELITY OVER A SYNTHETIC SCALAR FIELD

Travel Budget	Model Quality	Actual quality	Relative error
6.0	7.16	7.64	0.48
8.0	8.46	9.38	0.92

As a secondary and more intuitive measure of the quality of our method, we put a regular 6×6 node network fitted over the same field (Figure 9) and run the MIQP model such that we just have enough budget to obtain a full utility of 36. We let the start node be the second node from the left on the first row. From the output we can then

estimate values for all nodes that are not visited on the tour. We plot the much sparser survey data over the same space for time steps 100, 150, and 200 as shown in Figure 10. Comparing these figures with the corresponding ones from Figure 9, we observe that our models provide reasonable estimation of the entire synthetic scalar field without the need to visit all the nodes.

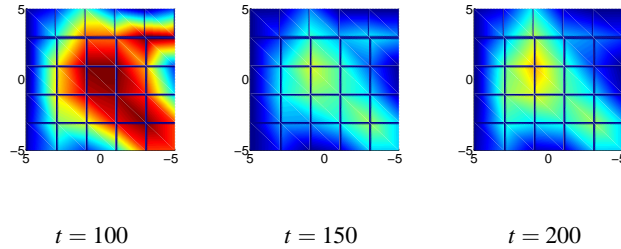


Fig. 10. Estimated scalar field based on the data obtained using the MIQP model.

B. Temperature Scalar Field

Our second simulation works with real temperature data retrieved from National Oceanographic Data Center⁴. The data consists of monthly average temperatures taken at 14 locations in the state of Massachusetts (see Figure 11) over a 24 month period. Using methodology similar to the synthetic scalar field example, we use the first year's data as training data (*i.e.*, $T = 12$) and then run our algorithm to sample and estimate temperature for the next year for every three months (a total of $12/3 = 4$ sets of temperatures). Node 10 (Boston) is selected as the base. The ground truth for these four sets is plotted in the top row of Figure 12.

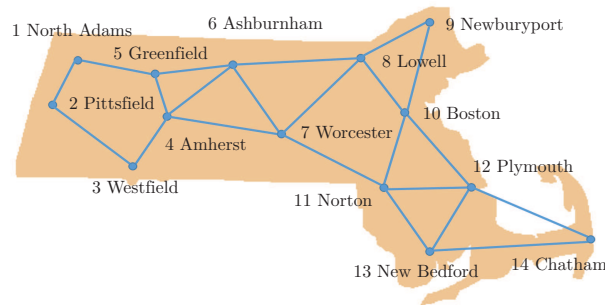


Fig. 11. Node network of 14 weather stations in Massachusetts.

For constructing the budget, since the area of Massachusetts is relatively small, we treat it as flat and use longitudes and latitudes to measure the distance between the nodes. We run our algorithm using varying budgets from 2.0 to 6.0. The results are plotted using heat map in Figure 12. Although the figures are visualizations of discrete data through interpolation, via similarity, we observe that additional budget allows better sampling and estimation quality. Quantitatively, since temperature itself is a good metric, we measure the quality of the estimation by summing up the absolute difference between actual and estimated values at each node. Then, the total error is averaged over the number of nodes. The outcome is listed in Table XI. At a budget of 3.0, which is enough to visit half of the nodes, the estimation quality is already fairly good at an average error of 0.27°C . The error goes to less than 0.1°C with budget of 6.0. On the other hand, visiting all nodes requires a budget of roughly 8.5.

VI. CONCLUSION AND FUTURE WORK

We introduced COP (and QCOP) as an extension to OP to address the problem of planning tours for surveying a spatially correlated field that also changes over time. Our computational experiments showed that the MIQP-based

⁴<http://www.nodc.noaa.gov/General/temperature.html>

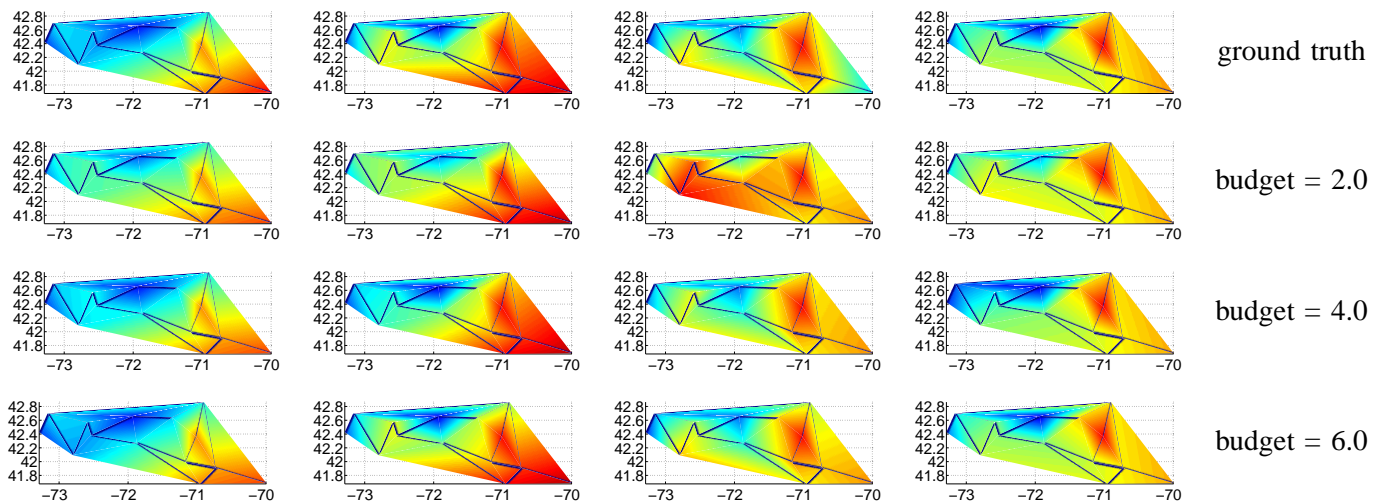


Fig. 12. Heat maps of Massachusetts in four different seasons of a year. Note that the color-scaling for each figure is different horizontally but the color-scaling used for each column of figures is the same. [top row] Interpolations using temperature data at 14 weather stations. [rows 2-4] Interpolations using estimated data with various budgets.

TABLE XI
TEMPERATURE ESTIMATION ERROR WITH RESPECT TO BUDGET

Travel Budget	2.0	3.0	4.0	5.0	6.0
Average Error ($^{\circ}\text{C}$)	0.46	0.27	0.22	0.15	0.08

anytime algorithms for QCOP are effective in capturing the spatial correlation among nearby nodes, indicating that QCOP and the associated MIQP models are applicable to persistent monitoring tasks in which the mobile robots have limited travel range.

For future work, we plan to improve our models to better capture real-world application scenarios. There are many promising directions; we mention two here. First, instead of looking at only immediate correlations as we did with QCOP, it may be beneficial to look at correlations of nodes that are further apart in the node network, *i.e.*, nodes that are in the 2- or 3-neighborhood. This extension is not trivial since the inclusion of additional nodes will certainly pose new computational challenges. We expect, however, the gain in estimation accuracy will degrade quickly as the neighborhood expands. Therefore, a 3-neighborhood is perhaps all that is needed. Second, for QCOP, the current weight selection criterion for applying the model in practice is somewhat ad-hoc and has ample room for improvement. Through a more systematic approach, perhaps via statistical methods, we hope to derive more principled and systematic procedures for selecting the weights for the MIQP models to further improve its applicability.

Furthermore, this paper only begins to address the problem of using correlations in informative path and policy planning in a discrete fashion. The dual problem to this estimation problem is a learning problem: how can we learn the correlations among the nodes so as to apply the methods from this paper? How can we carry out the learning task with limited number of mobile robots? Can we perform learning and estimation simultaneously? We will investigate these and other problems in the future.

REFERENCES

- P. Vansteenwegen, W. Souffriau, and D. V. Oudheusden, “The orienteering problem: A survey,” *European Journal of Operational Research*, vol. 209, pp. 1–10, 2011.
- G. Laporte, “The traveling salesman problem: An overview of exact and approximate algorithms,” *European Journal of Operational Research*, vol. 59, pp. 231–247, 1992.
- I. Chao, B. Golden, and E. Wasil, “Theory and methodology—the team orienteering problem,” *European Journal of Operational Research*, vol. 88, pp. 464–474, 1996.
- , “Theory and methodology - a fast and effective heuristic for the orienteering problem,” *European Journal of Operational Research*, vol. 88, pp. 475–489, 1996.
- R. M. Karp, “Reducibility among combinatorial problems,” in *Complexity of Computer Computations*, R. E. Miller and J. W. Thatcher, Eds. New York: Plenum, 1972, pp. 85–103.

- N. Michael, E. Stump, and K. Mohta, "Persistent surveillance with a team of MAVs," in *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2011, pp. 2708–2714.
- R. N. Smith, M. Schwager, S. L. Smith, B. H. Jones, D. Rus, and G. S. Sukhatme, "Persistent ocean monitoring with underwater gliders: Adapting sampling resolution," *Journal of Field Robotics*, vol. 28, no. 5, pp. 714–741, Sep-Oct 2011.
- S. Alamdari, E. Fata, and S. L. Smith, "Persistent monitoring in discrete environments: Minimizing the maximum weighted latency between observations," *International Journal of Robotics Research*, 2012, to appear.
- E. Arvelo, E. Kim, and N. C. Martins, "Memoryless control design for persistent surveillance under safety constraints," September 2012, available at <http://arxiv.org/abs/1209.5805>.
- C. G. Cassandras, X. Lin, and X. Ding, "An optimal control approach to the multi-agent persistent monitoring problem," *IEEE Transactions on Automatic Control*, vol. 58, no. 4, pp. 947–961, April 2013.
- A. Girard, A. Howell, and J. Hedrick, "Border patrol and surveillance missions using multiple unmanned air vehicles," in *Proc. 43rd IEEE Conference on Decision and Control*, 2004, pp. 620–625.
- B. Grocholsky, J. Keller, V. Kumar, and G. Pappas, "Cooperative air and ground surveillance," *IEEE Robotics and Automation Magazine*, vol. 13, no. 3, pp. 16–25, Sep 2006.
- X. Lan and M. Schwager, "Planning periodic persistent monitoring trajectories for sensing robots in Gaussian random fields," in *Proc. of the IEEE International Conference on Robotics and Automation (ICRA 13)*, May 2013, pp. 2407–2412.
- N. Nigam and I. Kroo, "Persistent surveillance using multiple unmanned air vehicles," in *Proc. IEEE Aerospace Conference*, 2008, pp. 1–14.
- J. L. Ny, M. A. Dahleh, E. Feron, and E. Frazzoli, "Continuous path planning for a data harvesting mobile server," in *47th IEEE Conference on Decision and Control*, 2008, pp. 1489–1494.
- S. L. Smith, M. Schwager, and D. Rus, "Persistent robotic tasks: Monitoring and sweeping in changing environments," *IEEE Transactions on Robotics*, vol. 28, no. 2, pp. 410–426, April 2012.
- D. E. Soltero, M. Schwager, and D. Rus, "Generating informative paths for persistent sensing in unknown environments," in *Proc. of the International Conference on Intelligent Robots and Systems (IROS 12)*, October 2012, pp. 2172–2179.
- G. A. Hollinger and G. S. Sukhatme, "Sampling-based robotic information gathering algorithms," *International Journal of Robotics Research*, vol. 33, no. 9, pp. 1271–1287, 2014.
- J. Yu, S. Karaman, and D. Rus, "Persistent monitoring of events with stochastic arrivals at multiple stations," in *Proceedings IEEE International Conference on Robotics & Automation*, 2014, pp. 5758–5765, preliminary extended version available at <http://arxiv.org/abs/1309.6041>.
- L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics & Automation*, vol. 12, no. 4, pp. 566–580, Jun. 1996.
- S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," Iowa State University, Tech. Rep., Oct 1998, computer Science Department TR 98-11.
- S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, June 2011. [Online]. Available: <http://ares.lids.mit.edu/papers/Karaman.Frazzoli.IJRR11.pdf>
- H. Choset, "Coverage of known spaces: The boustrophedon cellular decomposition," *Autonomous Robots*, vol. 9, pp. 247–253, 2000.
- , "Coverage for robotics—a survey of recent results," *Annals of Mathematics and Artificial Intelligence*, vol. 31, pp. 113 – 126, 2001.
- Y. Gabriely and E. Rimon, "Competitive on-line coverage of grid environments by a mobile robot," *Computational Geometry*, vol. 24, no. 3, pp. 197–224, 2003.
- W.-P. Chin and S. Ntafos, "Optimum watchman routes," *Information Processing Letters*, vol. 28, pp. 39–44, 1988.
- P. Hokayem, D. Stipanovic, and M. Spong, "On persistent coverage control," in *Proc. 46th IEEE Conference on Decision and Control*, 2008, pp. 6130–6135.
- S. Ntafos, "Watchman routes under limited visibility," *Computational Geometry*, vol. 1, pp. 149–170, 1991.
- J. Yu, M. Schwager, and D. Rus, "Correlated orienteering problem and its application to informative path planning for persistent monitoring tasks," in *Proceedings IEEE/RSJ International Conference on Intelligent Robots & Systems*, 2014.
- Gurobi Optimization Inc., "Gurobi optimizer reference manual," 2014. [Online]. Available: <http://www.gurobi.com>