STARS: Static Relays for Remote Sensing in Multirobot Real-Time Search and Monitoring

Yuanteng Pei, *Student Member*, *IEEE*, and Matt W. Mutka, *Fellow*, *IEEE*

Abstract—Mobile surveillance and sensing systems need a networking infrastructure that enables the mobile systems to transmit information gathered to a base station. We consider the problem of an efficient use of mobile robots to sense not only the region but also deploy relays to build the networking infrastructure. To develop an efficient solution to the above problem, we first present a problem called precedence constrained two traveling salesman (PC2TSP). We propose a near-optimal heuristic to PC2TSP to generate tours by clustering points, generating optimal single-traveler tours, and tour pruning and balance. By modeling in part by PC2TSP, we then solve the problem of minimum time two-robot real-time search with online relay deployment. We call the solution *STAtic Relay aided Search* (STARS), which identifies visiting positions, assigns the precedence constraint, and finally generates tours by PC2TSP. STARS enables solutions for remote robotic sensing and control. In addition, STARS substantially reduces cost compared to a homogeneous mobile robot system and enables constant monitoring of suspicious areas. STARS and our solution to PC2TSP are extensible to deal with more than two travelers. Extensive simulations show that our solution to PC2TSP achieves near-optimal performance with less than 2 percent average difference from optimal.

Index Terms—Multiple traveling salesman, precedence constraint, relay placement, coverage, motion planning, teleoperation, remote control

1 Introduction

Multirobot exploration of an unknown environment has been studied extensively in mobile robotics. In many applications of multirobot exploration, the robot's video and audio streams are transmitted back to the human operator at the base station (BS). In this paper, we study the problem of minimum time for multirobots to establish a network infrastructure that enables transmission of data back to a base station, while the multirobots also minimize the time to explore an area. To study this problem, we first consider the traveling salesman problem (TSP), which is a widely studied combinatorial optimization problem with extensive applications. A variation of TSP that we address here is the minimum time *precedence constrained two traveling salesman* (PC2TSP) from a given city. We first introduce the problem and then show its application.

PC2TSP can be stated as follows: Given 1) a finite set of cities $\mathbb{C} = \{1, 2, \ldots, n\}$ with each city marked as a consumer, a supplier, or both, and each consumer city has its supplier city set, 2) the cost c_{ij} of traveler's traveling time between each pair of cities $i, j \in \mathbb{C}$, c_{ij} and c_{ji} may differ, and 3) the precedence constraint that the traveler's arrival time plus waiting time at each consumer city is no earlier than the arrival time of all its supplier cities, to find two nonoverlapping traveler tours that allow traveler's nonzero waiting time at cities and both tours start from the base city 1 to visit each of the remaining cities exactly once, such that the maximum of the two tour times is minimized.

The problem is more challenging than other variants of multitraveling salesman problem (mTSP) in that a consumer city and some of its supplier cities may be visited by different travelers. Therefore, it is possible that a traveler arriving at a consumer city i must wait for the other traveler to visit i's supplier cities, introducing the dependence between travelers.

We apply PC2TSP to mobile surveillance systems via the problem of minimum time *two-robot real-time search with relay deployment* (2RSRD), which is described as follows: Given two robots, or mobile sensors, with ability of deploying relays as they search the area, how to compute the moving paths for robots and the proper time and positions to deploy relays along the movement path, so that 1) robots can efficiently search the area with minimum amount of time; and 2) video streams from robots at any sensing position (SP) can be successfully transmitted back to the base station. Fig. 1 shows the two robots in the search process. Any stream transmission at the sensing positions is relayed by a subset of two relays in the middle.

Similar to PC2TSP, 2RSRD implies a *precedence* constraint: When a robot conducts search and video transmission at sensing locations, supporting relays must be deployed *prior to* the robot's video transmission, so that streams can always be sent to the BS on a valid path.

Now that 2RSRD resembles PC2TSP, we model 2RSRD in part by PC2TSP as follows: The mobile robots become the travelers. The sensing positions for robots to conduct search become the consumer cities and the positions at which relays are deployed become the supplier cities. For each sensing position modeled as a consumer city, its

- *Y. Pei is with Platform and Infrastructure Engineering Division, eBay Inc., 2121 North 1st Street, San Jose CA 95037. E-mail: ypei@ebay.com.*
- *M.W. Mutka is with the Department of Computer Science and Engineering, Michigan State University, 3115 Engineering Building, East Lansing, MI 48824-1226. E-mail: mutka@cse.msu.edu.*

Manuscript received 4 Dec. 2011; revised 1 Oct. 2012; accepted 9 Oct. 2012; published online 19 Oct. 2012.
Recommended for acceptance by D. Simplot-Ryl.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number TPDS-2011-12-0882.
Digital Object Identifier no. 10.1109/TPDS.2012.299.

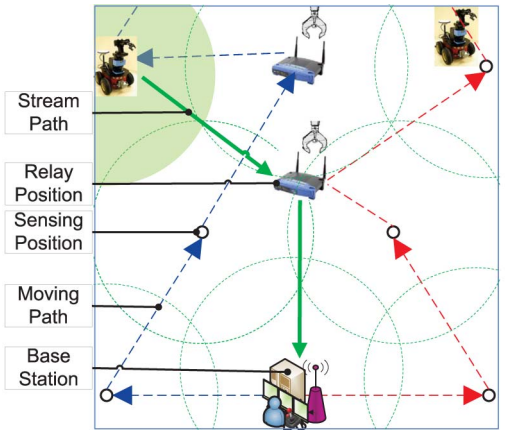


Fig. 1. Two robots search the environment and deploy static relays as they travel along the paths. The left robot transmits streams to the base station via the center relay deployed by the right robot.

corresponding supplier city set is all of its supporting relay positions (RP) that form a valid path to the BS. Therefore, as long as we have 1) identified sensing and relay positions, and 2) assigned the precedence constraint that choose the relay set for each sensing position as a valid path to BS, we can use PC2TSP to solve 2RSRD as the final step to generate tours.

By modeling in part by PC2TSP, we present our solution called *STAtic Relay aided Search* (STARS) to solve 2RSRD. STARS enables remote robotic sensing and control. It has extensive applications such as reconnaissance, surveillance, search and rescue missions in dangerous and hostile areas. STARS substantially reduces cost by replacing expensive robots by low cost static relays, compared to a homogeneous mobile robot system where robots have to be the relays to transmit streams. Another benefit is that STARS also enables the deployment of static sensors to monitor the "suspicious areas" where the human operator and the computer system decide that the targets may (re)appear after the robots migrate away. The details of how STARS supports constant monitoring by static sensors is described in Section 6.2.

With this heterogeneous system of mobile robots, static sensors, and relays, STARS has the merits of both static sensors and mobile robots. Sensors are relatively inexpensive devices that are good at constantly monitoring regions, but they require a large quantities in deployment to cover the area. Robots are more powerful to perform complex tasks but are markedly more expensive. They are better suited for situations requiring a single time search and exploration process. Compared with homogeneous static sensor networks, the proposed method saves unnecessary sensors because sensors only need to be placed as the suspicious area rather than the whole area. Compared with homogeneous mobile robot networks, our scheme has notably less cost but can monitor suspicious area constantly thanks to the deployed sensors.

To the best of our knowledge, STARS is the first of its kind to leverage online relay deployment as a communication backbone to support real-time communication, remote control and sensing with mobile robots and static sensors. There are many challenges, which are discussed in Section 3.1.

1.1 Main Contribution

- We first present a problem called precedence constrained two traveling salesman. We propose a near-optimal heuristic to PC2TSP to generate tours by clustering points, generating optimal single-traveler tours, and tour pruning and balance.
- To solve PC2TSP, we propose the tour validation and dependence checking scheme to detect deadlock and self-flipped cases that are invalid. It also checks the dependence on two tours and adds necessary waiting time.
- To solve 2RSRD, we propose STARS, which divides the problem into two subproblems: 1) preprocessing: identify visiting positions and assign the precedence constraint, which are modeled by set cover, Steiner connected dominating set, and breadth-first search; 2) tour generation, which is modeled by PC2TSP.

To maintain consistency and show a valid application of PC2TSP, we present the solution of PC2TSP in the context of STARS as tour generation. However, it can be used as a stand-alone solution to PC2TSP with minor changes given in Section 5. The remainder of the paper proceeds as follows: Section 2 introduces the system model. In Section 3, we formulate the problem and discuss the challenges. Sections 4 and 5 give the preprocessing and tour generation. Then, several extensions are discussed in Section 6. Performance evaluation is in Section 7 and related work is summarized in Section 8. Section 9 concludes the paper and discusses the future work.

2 System Model

Environment model. We use a 2D occupancy map to model the environment as an unsearched rectangular region $\mathbb{R} = X \times Y$ with grid cells similar to that in [1]. The cell size equals to the footprint size of the robot. A cell status includes searched, unsearched, and obstructed. An obstacle cell cannot be visited by the robots nor can a relay be deployed upon it. Obstacles do not affect communication but block the sensing range. We assume the obstacle distribution in the environment is known. We would like to tackle the unknown or inaccurate obstacle distribution in the future work, where the tour is recomputed by updating the new data of obstacles as input. The *base station* is the control center where the human operator can remotely monitor and potentially operate the robots if necessary.

Robot model. The number of robots in our model is limited to two because the goal is to use low cost static relays to replace high cost mobile robots. The *two robots* are denoted as a and b. The solution is extensible to more than two robots, which will be discussed in Section 6. The robots move at a constant speed where the acceleration and deceleration time is not considered. Each robot has traveling, communication, sensing, and relay deployment capabilities. It scans the environment using cameras and acoustic sensors with sensing range r_s and communicates with other nodes with communication range r_c by a 802.11

radio, where $r_c \geq r_s$. We denote the communication to sensing range ratio as $\boldsymbol{\theta = r_c/r_s}$. Since only two robots are sending streams, we assume bandwidth is adequate for transmission. The unit disk model is used for sensing and communication. As robot's sensing devices, such as laser finder and cameras, are more powerful than those in limited capability sensors, the sensing range is more stable. A more sophisticated model may not be necessary.

Search with relay deployment model. The robots search the region by visiting a *sensing position*, where the robots stop and enter the *search and transmit interval* (STI) and the unsearched cells in its covered area with radius r_s will be marked as searched. The STI is for the robots to sense the area and send video streams back to the BS. The ratio of searched cells out of total nonobstacle cells is denoted as $\boldsymbol{\theta_s}$. The *completion* of search is defined by $\theta_s \geq \delta_s$, where $\boldsymbol{\delta_s}$ is a *search complete threshold*, for example, 99 percent. $\mathbb{SIP}$ is the set of SPs whose visit complete the search. $\mathbb{SIP}$ includes the BS.

The robots deploy relays by visiting a *relay position* in the search tour. A relay is a nonmobile device with 802.11 for transmitting streams from a robot with the same communication range r_c. A robot has a maximum carrying capacity of n_r relays. We first assume STI and relay deploying time are negligible. The model is easily extensible to consider a nontrivial STI and relay deploying time, which will be discussed in Section 6.

SP(a) and SP(b) denote the vector of SPs that robot a and b need to visit; RP(a) and RP(b) denote the vector of RPs a and b need to visit. $\boldsymbol{P_a}$ and $\boldsymbol{P_b}$ denote the ordered vector of sensing and relay positions a and b traverse. If we convert P_a and P_b to unordered set P'_a, P'_b, $P'_a = SP(a) \cup RP(a)$, $P'_b = SP(b) \cup RP(b)$. $\boldsymbol{T_a}$ and $\boldsymbol{T_b}$ denote the *tour* a and b traverse, or the vector of position and the waiting time w at each position: $T_a = \{P_a, w(P_a)\}$, $T_b = \{P_b, w(P_b)\}$. A *path* denoted by path(V) gives a route with an ordered vector of positions of $\mathrm{V} = (v_0), v_1, \ldots, v_n)$. $\boldsymbol{t_a}$ and $\boldsymbol{t_b}$ give the tour time in seconds, or the total time needed to traverse all positions in P_a or P_b including the possible waiting time. $\boldsymbol{t(i)}$ gives the start of search time for position i.

We compute the tour off line with the information of the environment before the actual search starts. Robots are not modeled to come back to the BS immediately after the search is complete because they should remain in the frontier area for remote sensing.

3 Problem Formulation

Given the system model with notations defined in Table 1, we formulate the problem and explain the basic idea.

Minimum time *two-robot real-time search with relay deployment* (2RSRD): Given a region $\mathbb{R}$, two robots a and b each with relay carrying capacity of n_r, compute tours T_a, T_b to obtain the minimum search time t_{ab} such that:

$$\text{Minimize } t_{ab} = \max\{t_a, t_b\} \tag{1}$$

$$\text{Subject to } \mathbb{SIP} = SP(a) \cup SP(b), \tag{2}$$

$$\exists path(V), V = (BS, rp_0, \ldots, rp_n, i), \quad \forall i \in \mathbb{SIP}, \tag{3}$$

TABLE 1
Notations in STARS

Notations	Definitions
PC2TSP	Precedence constrained two traveling salesman: the theoretical problem.
2RSRD	Two-robot real-time search with relay deployment: the robotic search problem.
STARS	STAtic Relay-aided Search: our solution to 2RSRD, where the tour generation subproblem is modeled by PC2TSP.
$\mathbb{R}$	Search region.
$\mathbb{P}$	Precedence constraint matrix.
SP, RP, $\mathbb{SP}, \mathbb{RP}$	Sensing position, relay position; their set.
a, b	The two mobile robots.
BS	Base station.
P_a, P_b	The ordered vector of sensing and relay positions a and b traverse.
T_a, T_b	Tours of robot a and b.
t_a, t_b	Tour times in seconds of robot a and b.
t_{rd}	Relay deploying time at the relay positions.
$t(i)$	The start of search time for position i.
STI	Search and transmit interval when robots stops at the sensing positions.
S-CDS	Steiner connected dominating set.
n_r	Robot's relay carrying capacity.
$dist(i, j)$	Distance between i and j.
r_c, r_s, θ	Communication and sensing range, and their ratio.

$$dist(v_j, v_{j+1}) \leq r_c, \quad \forall v_j \in V, j < n, \tag{4}$$

$$t(rp_j) \leq t(i), \quad \forall j \in 0, \ldots, n, \tag{5}$$

$$|RP(a)| \leq n_r, \quad |RP(b)| \leq n_r, \tag{6}$$

where rp_i is a relay position in the path ($rp_i \in RP(a) \cup RP(b)$). Equation (2) indicates the search is complete. Equations (3), (4), and (5) give that for each sensing position i in a or b's tours, there is a valid path back to the BS at the time of $t(i)$. Equation (6) gives the carrying capacity constraint.

3.1 Challenges

We first prove 2RSRD is NP-hard and then show other challenges to directly solve the problem.

Theorem 1. *2RSRD is NP-hard.*

Proof. We give a polynomial time Turing reduction from the single *precedence constrained asymmetric traveling salesman problem* (PCATSP) to 2RSRD. That is, we present an algorithm A' that solves PCATSP by using an algorithm A for solving 2RSRD as a subroutine. Given an algorithm A for solving 2RSRD with an input of a region R. R has a vertical line composed of obstacle cells in the middle of the region, separating R into two identical subregions. The two robots will only travel inside the subregion due to the obstacles. An algorithm A' can solve PCATSP for the subregion by making a single call to the subroutine A on the region R. If there is a polynomial time algorithm A for solving 2RSRD, we will have a polynomial time algorithm A' for solving PCATSP. It contradicts to the fact that PCATSP is NP-hard [2], and this completes the proof. □

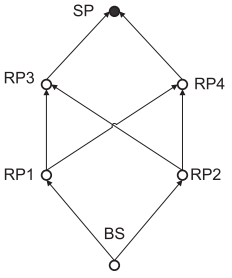


Fig. 2. Multiple choices of precedence constraints.

There are other challenges to solve 2RSRD directly other than its NP-hardness. First, the robots may need to wait for each other if one relies on the other's not deployed relay to transmit streams. There is a dependence between both robots: One may have to wait for the other to deploy the supporting relays. The dependence and waiting time make it challenging to directly apply linear programming formulation to solve the problem.

Second, search time t_{ab} is affected by the robot's relay carrying capacity and whether to have a goal of reducing relay size. Suppose that relays are sufficient, carrying capacity is infinite, and robots place a relay at every SP. As long as robots travel on consecutive SPs, the stream transmission requirement is always met (recall $r_c \geq r_s$). The relay requirement will have little impact on the tour generation. The problem is then simplified and less interesting to solve. On the other hand, if relays are only placed selectively (such as at only S-CDS positions as in Section 4.2), the tour computation is constrained by visiting the relay positions first before visiting other sensing positions that are supported by these relays.

Third, the precedence constraint is *uncertain* compared with that in PCATSP, where position i preceding j is explicitly stated. 2RSRD requires that *at least* one valid relay path has been deployed at the searching time on any SP. However, multiple sets of relays can fulfil this requirement. To illustrate, there are four RPs as shown in Fig. 2. When search starts at SP, at least one pair of RPs (1,3), (2,4), (1,4), (2,3) is required to be deployed. The uncertainty makes the problem more difficult to deal with.

Therefore, to solve 2RSRD optimally, which is to find *all possible*:

1. SPs that complete coverage,
2. RPs that cover all SPs (which is a S-CDS as we define below),
3. precedence constraints that ensure the stream relay requirement,
4. partitioning of visiting positions,
5. permutation of tours, and
6. adding waiting time for each pair of tours when dependence occurs.

3.2 STARS: STAtic Relay Aided Search

Due to the problem's high complexity, we propose a high-quality heuristic called STARS. It divides the problem into two subproblems: 1) preprocessing: identify visiting positions and assign the precedence constraint, 2) tour generation, which is modeled by PC2TSP.

We present a valid solution for the first subproblem. We also show how our near-optimal solution to PC2TSP is applied to solve the second subproblem of tour generation. Such division separates the key tour generation problem from a specific region with different visiting position selections, precedence constraints and robot carrying capacities. In future work, the tour generation is important to evaluate different schemes of selecting visiting positions and precedence constraints, making it possible to find the optimal visiting positions and precedence constraint.

The formulation for each part of the subproblems is as follows:

Problem 1a: Identify sensing positions. Given a region $\mathbb{R}$, how to identify a minimum number of sensing positions $\mathbb{SP}$ that completes the search (coverage requirement: searched cell ratio $\theta_s \geq \delta_s$):

$$\begin{aligned} &\text{Minimize } |\mathbb{SP}| \\ &\text{Subject to } \bigcup_{i \in \mathbb{SP}} cell_{searched}(i) \geq \delta_s \cdot |cell_{nonobstacle\,\mathbb{R}}|. \end{aligned} \tag{7}$$

Problem 1b: Identify relay positions. As the robots visit all sensing positions, the area will be fully searched. However, the video streams cannot be transmitted back to the BS unless a path formed by relays always exists. $\mathbb{RP}$ forms a communication *backbone*. Without worrying about the visiting sequence at this step, the backbone must at least "cover" all $\mathbb{SP}$ including the BS. Then from any sensing position there is a valid path back to the BS to transmit the streams (denoted by path (V) in (8)). We assume the user would like to save relays and with such an attempt the carrying capacity constraint is satisfied (satisfying (6)). An extension to take advantage of extra relays under the carrying capacity (difference of carrying capacity and the computed minimum number of relays) is presented in the supplementary file, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPDS.2012.299. Hence, the problem is: given $\mathbb{R}$ and $\mathbb{SP}$, how to identify a minimum number of relay positions $\mathbb{RP}$ that satisfies 1) each SP (denoted by i below) is either also in $\mathbb{RP}$ or a neighbor of RP; 2) $\mathbb{RP}$ includes the BS, and $\mathbb{RP}$ is connected:

$$\begin{aligned} &\text{Minimize } |\mathbb{RP}|, \\ &\text{s.t. } i \in \mathbb{RP} \| dist(i,j) < r_c, \quad \forall i \in \mathbb{SP}, \exists j \in \mathbb{RP}, \\ &\quad \exists \text{ path}(V), V = (v_0, \ldots, v_{n-1}), \quad \forall v_0 \in \mathbb{SP}, \\ &\quad dist(v_k, v_{k+1}) \leq r_c, \quad \forall v_k \in V, k < |V| - 1, \\ &\quad v_{n-1} = \text{BS}, \text{BS} \in \mathbb{RP}. \end{aligned} \tag{8}$$

In addition, a simple extension to leverage the preexisting relays is presented in the online supplemental material.

Problem 1c: Assign the precedence constraint. Given $\mathbb{R}$, $\mathbb{SP}$, and $\mathbb{RP}$, how to define a *precedence constraint matrix* $\mathbb{P}$: $\boldsymbol{p_{ij}}$, $\forall i, j \in \mathbb{SP} \cup \mathbb{RP}$:

$$p_{ij} = \begin{cases} 1, & \text{position } i \text{ precedes } j, \\ 0, & \text{otherwise.} \end{cases}$$

Validity. Matrix $\mathbb{P}$ is *valid* if and only if any tours on $\mathbb{SP}, \mathbb{RP}$ satisfying $\mathbb{P}$ meet the constraints defined in (3), (4), and (5).

Problem 2: Tour generation. Given $\mathbb{R}$, $\mathbb{SIP}$, $\mathbb{RIP}$, how to find two minimum time tours T_a, T_b that satisfy $\mathbb{P}$:

$$\begin{aligned} &\text{Minimize } t_{ab} = \max\{t_a, t_b\}, \\ &\text{Subject to } P_a \cup P_b = \mathbb{SIP} \cup \mathbb{RIP}, \\ &\qquad t(i) \le t(j), \quad \forall p_{ij} = 1. \end{aligned} \tag{9}$$

Rationale. Problem 1a is formulated by minimum $|\mathbb{SIP}|$ as a heuristic because less visiting positions may lead to a shorter total tour. Problem 1b is formulated to reduce the number of relays. Problem 1c is to remove the uncertainty of precedence constraints for the tour generation. With the result of problems 1a, 1b, 1c as input, the problem 2 is formulated to generate the tours.

4 Preprocessing

4.1 Identify Sensing Positions

Recall the problem formulation 1a in Section 3.2. We formulate the problem as the well-known NP-hard *set cover* problem, which is to select a minimum number of sets so that the union of the sets contains all the elements in the universe. Here, the search region is divided into grid cells. With each cell as an element, all the nonobstacle cells become the universe. Each SP's sensing area is a set that contains some elements within sensing range, r_s. Since the problem is NP-hard, we leverage a heuristic modified from the *triangle lattice pattern* (TLP), which is proven optimal in terms of the number of sensing disks to obtain full coverage [3] if the region is free of obstacle. If the TLP chosen positions are the obstacle cells which cannot be placed sensors on, we greedily select from nonobstacle positions to cover the remaining uncovered area. Algorithm 1 describes the procedure.

Algorithm 1. Identify sensing positions for full region coverage

1 **Input:** A region $\mathbb{R}$. **Output:** $\mathbb{SIP}$.
2 Follow the triangle lattice pattern to place $\mathbb{SIP}$ on nonobstacle cells.
3 **While** $\theta_s < \delta_s$ **do** greedily select a nonobstacle position i to cover a maximum number of uncovered cells. Insert i into $\mathbb{SIP}$.
4 With an equal coverage, the position with minimum distance to the nearest SP neighbor is selected.

4.2 Identify Relay Positions

Recall the problem formulation 1b in Section 3.2. We model the problem as the minimum *Steiner connected dominating set* (S-CDS) problem.

Connected dominating set (CDS) problem is a classical problem in combinatorial optimization with the following definition: Given a graph $G = (V, E)$, find the smallest subset S of vertices that induce a connected subgraph and each vertex in $V - S$ is adjacent to at least one vertex in S. S-CDS is a generalization of CDS where only a specified set $R \subseteq V$ of required vertices has to be dominated by a connected dominating set [4]. Given all nonobstacle cells in $\mathbb{R}$ as V, $\mathbb{RIP}$ only needs to cover $\mathbb{SIP}$, which is a subset of V. We require $\mathbb{RIP}$ induce a connected subgraph because streams from any vertex in $\mathbb{SIP}$ is sent to the BS.

Differently from those in [1] and [5], this relay placement problem is not modeled as the Steiner minimum tree with the minimum Steiner point (SMT-MSP) problem. The reason is that a terminal point can always also serve as a Steiner point to connect to other terminal points in SMT-MSP. However, in our application, the robots only pass the sensing positions once. A sensing position (terminal point) cannot serve as a Steiner point (relay) unless a real relay is deployed.

The basic idea is simple. First find the minimum dominating set which may not be connected, then add vertices to connect the dominating set. The details of the algorithm are shown in Algorithm 2. The algorithm is similar to the Steiner-CDS algorithm in [4]. The three major differences are: First, we use an optimal dominating set (DS) finder (finding DS is NP-hard) when the size is smaller than a threshold $\delta_{ds} = 70$ and use the greedy heuristic as in [4] when the input size is larger. The DS finder can compute up to 70 vertices to optimal in less than 1 minute on a regular notebook computer, improving the S-CDS quality. The threshold is set as 70 to tradeoff the computation efficiency and scale. **DS** is the overlapping set: $\text{DS} = \mathbb{SIP} \cap \mathbb{RIP}$.

Algorithm 2: Steiner-CDS algorithm to identify relay positions

1 **Input:** $\mathbb{SP}$. **Output:** $\mathbb{RP}$.
// 1. Obtain the dominating set (DS) on $\mathbb{SP}$.
2 **if** $|\mathbb{SP}| < \delta_{ds}$ **then**
3 Obtain DS by the optimal DS finder.
4 **else**
5 Obtain DS by the greedy set cover with maximum nodes.
// 2. Apply Steiner tree approximation algorithm to connect DS.
6 Call *Steinerized Minimum Spanning Tree* [6] to connect DS. The newly added nodes are RP_2
7 Return $\mathbb{RP} = DS \cup RP_2$.

The second difference is that RP_2 are placed on the shortest obstacle-aware paths computed by A* search between its end vertices when necessary to avoid obstacles. A* search guarantees the same optimal result as Dijkstra but is more efficient. Because it reduces the searching space compared to BFS using an admissible and consistent heuristic function which does not overestimate the distance to the goal [1]. Third, we always include the BS in DS because the BS belongs to the backbone: $BS \in \mathbb{RIP}$.

4.3 Assign the Precedence Constraint

Recall the problem formulation 1c in Section 3.2. We first define notations, then present our solution and prove its validity. We define the *pure sensing positions* $\mathbb{SIP}_{pure} = \mathbb{SIP} \backslash \text{DS}$ ("\" means set subtraction), and *visiting positions* $\mathbb{VIP} = \mathbb{SIP} \cup \mathbb{RIP}$.

Algorithm 3 shows the procedure of assigning the precedence constraint by first assigning a *single* parentID for each node in $\mathbb{VIP}$. Assigning parentIDs is important because we can always backtrack a node's parentID recursively to the BS, forming a valid path for communication. The procedure builds a precedence tree to find parentIDs on $\mathbb{VIP}$. The tree has 1) the nonleaf backbone as $\mathbb{RIP}$, 2) leaf nodes as $\mathbb{SIP}_{pure}$, and 3) BS as the root.

Algorithm 3: Assign parentIDs and the precedence constraint

Input: $\mathbb{SP}, \mathbb{RP}$. **Output:** $\mathbb{P}$.
// 1. Assign parentIDs. The BS has an invalid parentID(-1).
Breadth-first search on $\mathbb{RP}$ to obtain the *RPlevel* in BFS. Assign a parentID in BFS for each relay according to BFS queueing order.
For each $i \in \mathbb{SP}_{pure}$, insert all i's 1-hop RP neighbors into its parent candidate set.
Loop through $i \in \mathbb{SP}_{pure}$, **if** *i has a single parent candidate j* **then**
  i.parentID$\leftarrow j$
else
  i.parentID$\leftarrow$ minimum *RPlevel* node j.
j.chilren.push_back(i). End loop.
// 2. Assign values in the precedence matrix $\mathbb{P}$
For each $j \in \mathbb{SP}$, get ancestors of j by recursively backtracking its parentID until root. Set $p_{ij} = 1$ when i is an ancestor of j.

Theorem 2. *The precedence constraint matrix $\mathbb{P}$ obtained by Algorithm 3 is valid.*

Proof. Each node in $\mathbb{WP}$ has a single parentID. Because $\mathbb{SP} \subseteq \mathbb{WP}$, each node in $i \in \mathbb{SP}$ has a single parentID. For each i, its ancestors form a valid path back to the root BS, being part of the backbone. Because $\mathbb{P}$ is defined that all ancestors of i are required to be visited before visiting i, $\mathbb{P}$ is valid, and this completes the proof. □

5 Tour Generation by PC2TSP

Recall the problem formulation 2 in Section 3.2. We model the problem as PC2TSP. By a similar proof of Theorem 1 we can prove PC2TSP is NP-hard. Therefore, we provide a near-optimal heuristic that 1) clusters the nodes (with possible overlapping of shared relays) into two parts, 2) solves them individually by PCATSP optimally, and 3) prunes the shared relays and balances the two tours.

The tour generation procedure can be used as a standalone solution to the theoretical problem of PC2TSP outside of the robotics domain. The following changes are needed: 1) Change robots, sensing positions, relay positions, and the BS to travelers, consumer cities, supplier cities, and the base city. 2) Change ones ancestors and parentID to its supplier cities in tour validation and dependence checking. 3) Values in the cost matrix c_{ij} and whether to have the conversion from Hamiltonian path depend on the application.

5.1 Cluster Visiting Positions

We first cluster each node $i \in \mathbb{SP}$ according to its x-coordinate and insert it accordingly to either P_a or P_b: assign to robot a or b by whether the position is on the left or right of the center line. A *center line* $X = x_{\mathbb{R}}/2$ is a vertical line to partition the region in the middle. Then, the supporting relays, or the ancestors of each $i \in \mathbb{SP}$ are assigned accordingly to the same cluster as i. Therefore, position sets P_a and P_b include the sensing positions and their supporting relays. Relays may be (partially) shared by P_a and P_b. The motivation of such clustering is to make a tour *self-reliant* without need of waiting for relays from the other tour.

We have tried more advanced schemes such as K-means and also attempted to balance the two sets to have similar amount of nodes such that $||P_a| - |P_b|| \leq 1$, however they do not improve the performance. We will leave the further investigation on other clustering algorithms and their impact in the future work. Although the initial P_a and P_b may be unbalanced, our tour pruning and balance in Section 5.3 work effectively in balancing the tours.

5.2 Obtain Optimal Single Robot Tour

Given the positions P_a and P_b for robot a and b to traverse, and the precedence constraint in $\mathbb{P}$ relevant to each part, we need to compute the precedence constrained shortest Hamiltonian path starting at BS for a and b individually. We first convert the Hamiltonian path problem to the traveling salesman problem, then model the problem as (single) PCATSP [7].

Conversion. In our system model, robots will not immediately return to the BS after the search completes. Therefore, the shortest Hamiltonian path problem with a given starting city should be first converted to an asymmetric TSP by setting the distance on the arcs from all vertices to BS as zero [8]. It forces the evaluation of all possible paths starting with BS and disregards the way back to the BS in the tour.

Mathematically, precedence constrained asymmetric (single) traveling salesman problem (PCATSP) is as follows:

$$\begin{aligned}
&\text{Minimize } \sum_{i=1}^{n} \sum_{j=1, j\neq i}^{n} c_{ij} x_{ij}, \\
&\text{Subject to } \sum_{i=1, i\neq j}^{n} x_{ij} = 1, \quad \forall j = 1, \ldots, n, \\
&\qquad \sum_{j=1, j\neq i}^{n} x_{ij} = 1, \quad \forall i = 1, \ldots, n, \\
&\qquad p_{ij} \geq x_{ij}, \quad \forall i, j = 2, \ldots, n, \quad i \neq j, \\
&\qquad p_{ij} + p_{ji} = 1, \quad \forall i, j = 2, \ldots, n, \quad i \neq j, \\
&\qquad p_{ij} + p_{jk} + p_{ki} \leq 2, \quad \forall i, j, k = 2, \ldots, n, \quad i \neq j \neq k, \\
&\qquad p_{ij} = 1, \quad \forall j = 2, \ldots, n, \quad \forall i \in SPC_j,
\end{aligned} \tag{10}$$

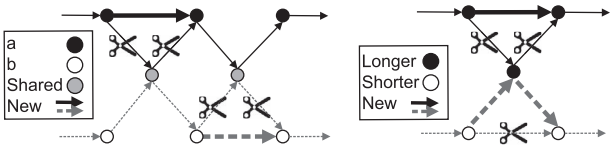
where c_{ij} is the traverse time from i to j based on the obstacle aware path length by A* search and a traverse velocity. c_{ij} forms the *cost matrix* $\mathbb{C}$. x_{ij} and p_{ij} are binary and $x_{ij} = 1$ indicates position i precedes j immediately. In the two constraints for x_{ij}, the first one is a constraint for outgoing link (for any node i, there cannot be more than one arc going out from i). Similarly, the other constraint x_{ij} is defined for the incoming link. $p_{ij} = 1$ means position i precedes j not necessarily immediately in the tour (p_{ij} here is the same as the one in Problem 1c in Section 3.2). SPC_j gives the subset of positions in $\{2, \ldots, n\}$ that are required to precede j.

Solving PCATSP by branch and cut method with the above integer linear programming formulation is relatively fast for a reasonable size despite its NP-hardness. Real-world instances with 200 nodes PCATSP have been solved optimally in a few minutes [9].

5.3 Prune and Balance Tours

5.3.1 Pruning Shared Relays

Tour balance is to minimize the total search time t_{ab}, which is dominated by the longer tour. The two paths by PCATSP



(a) Shared relay pruning. (b) Vertex transfer.

Fig. 3. (a) Shared relay pruning: duplicated relays (gray nodes) are divided in each robot's tour to reduce path length. (b) Vertex transfer: the center black node is transferred from the longer tour to the shorter one.

are self-reliant but may have visited duplicate relay positions. The goal of path pruning is to reduce search time t_{ab} by removing duplicate relay positions from the single robot tours in the previous step, considering the possibility and cost of introducing dependence with waiting time.

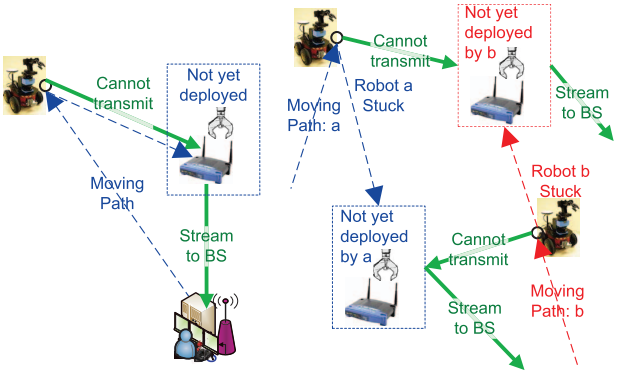
The pruning strategy has shown in Algorithm 4. With $|SR|$ number of shared relays, we try all possible ways of allocating them to robot a and b by computing the powerset. (A powerset of a set S is the set of all subsets of S.) Due to the exponential nature, we set the maximum segment length $k = 12$ and to divide SR into segments to limit the powerset size. For each segment, we compute the best allocation of new tour cost by first computing the time based on paths from c_{ij}. Then the dependence checking is called in line 7 to check whether dependence occurs and to add waiting time, if it exists. Fig. 3a illustrates the idea. Denote the two shared relays as SR 1 and 2. They are divided by each tour evenly after evaluating $PS(\{1,2\}) = \{\emptyset, \{1\}, \{2\}, \{1,2\}\}$.

Algorithm 4: Path pruning

1 **Input:** T_a, T_b. **Output:** updated T_a, T_b.
// Obtain shared relays and divide into segments if $|SR| > k$
2 Obtain shared relays SR from T_a, T_b in order. Divide it into $\lceil |SR|/k \rceil$ segments, each with k nodes (except the last).
3 **foreach** *segment i* **do**
4 Get powerset of i: $PS(i)$, $|PS(i)| \leq 2^k$
5 **foreach** $j \in PS(i)$, $j' = i \backslash j$ **do**
6 Get temp tours by $T'_a = T_a \backslash j$, $T'_b = T_b \backslash j'$. Update search time.
7 Call Alg. 5 to check dependency and obtain the true time t'_{ab} by adding wait time (if it exists) on T'_a, T'_b.
8 Save j_{min}, j'_{min} with minimum t'_{ab} and then update tours: $T_a = T_a \backslash j_{min}$, $T_b = T_b \backslash j'_{min}$.

Validate tours and check dependence. With the path positions P_a, P_b, we needs to verify whether they are feasible by checking whether self-flipped and deadlock cases occur. Then, we compute the true search time considering dependence.

Definitions. A tour is *self-flipped* if and only if it has a position i whose ancestors are visited after i. A self-flipped tour is invalid because by itself it violates the precedence constraint. A position i is *blocked* when it has its parent not visited at time $t(i)$. Position i has a missingID, which is set as i's parentID. Two tours T_a, T_b are in *deadlock* status when there exist $i \in T_a, j \in T_b$ whose missingIDs are after i, j in the other tour: missingID(i) in T_b after j, missingID(j) in T_a after i. A deadlock tour is invalid because both robots cannot proceed. Fig. 4 illustrates the two invalid cases. Fig. 4a shows the robot arrives at a SP that requires a relay not yet



(a) Self-flipped. (b) Deadlock.

Fig. 4. Invalid cases.

deployed. Fig. 4b shows robot a needs the relay to be deployed by b and vice versa.

The basic idea is to traverse T_a, T_b again and one tour is blocked when the needed relay is not yet visited in the other tour. At each position i, the tour time t is updated not only from the cost matrix c_{ij}, but also considers the waiting time for i's parent: update t as the maximum of its parent's visited time and itself. Note that when a parent is visited, all ancestors must have been visited, because an unvisited parent will immediately block its children. In addition, when self-flipped and deadlock cases occur, we mark the tours invalid and set the tour time to be a very large number, i.e., FLT_MAX, making the tour not selected in its parent function. The detailed procedure is shown in Algorithm 5.

Algorithm 5: Tour validation and dependency checking

1 **Input:** Local copies of any tours T_a, T_b. **Output:** t_{ab}, updated T_a, T_b with waiting time, isValid.
2 isDeadlock=false, isValid=true.
3 **if** *isFlipped(T_a)* || *isFlipped(T_b)* **then** return t_{ab}=FLT_MAX, isValid=false.
4 Iterators it_a, it_b at the beginning of T_a, T_b. $t_a = 0, t_b = 0$
5 **repeat**
6 **repeat**
7 it_a keeps on proceeding to the next position $++it_a$ in T_a. At each new position it_a, first to update time by c_{ij}: $t_a += c\{*(it_a - 1), *it_a\}$, and then make sure the robot waits for its parent: $t_a = \max(t_a, t(parentID(*it_a)))$, update wait time vector $w(P_a)$. update the visited time at $t(it_a) = t_a$, mark it_a as visited.
8 **until** *it_a is blocked when its parent is not visited.*
9 Same repeat procedure as above for it_b, which keeps on proceeding until it is blocked.
10 **if** *Both it_a, it_b are blocked* **then** isDeadlock=*checkDeadlock*(it_a, it_b).
11 **until** *it_a, it_b reach the end of T_a, T_b || isDeadlock*
12 **if** *isDeadlock* **then** return t_{ab}=FLT_MAX, isValid=false.

5.3.2 Balance with Vertex Transfer

While pruning is only effective on shared relays, vertex transfer attempts on any node and further balances tours to reduce t_{ab}. The basic idea is to try transferring a position on the longer tour and inserting it into the shorter one. We greedily choose the best pair of removal and insertion position until no further reduction is found. For each pair of newly modified tours, Algorithm 5 is called to add waiting time when it is applicable. The detailed procedure is

presented in Algorithm 6. As illustrated in Fig. 3b, the black visiting position in the middle is transferred to the shorter white one, where black positions are those on the longer tour and the white ones are those on the shorter tour.

Algorithm 6: Balance with vertex transfer

```
Input: T_a, T_b. Output: updated T_a, T_b.
repeat
    Mark the longer and shorter tours of T_a, T_b as T_l, T_s.
    foreach i ∈ T_l do
        foreach j ∈ T_s, j ≠ i do
            Get new tours T'_l ← (T_l with removed i),
            T'_s ← (T_s with inserted i after position of j).
            Call Alg. 5 to validate tours and check
            dependency. Obtain the true time t'_ls by adding
            wait time (if it exists) on T'_l, T'_s.
    Get min t'_ab_min from all t'_ab, record corresponding
    i'_min, j'_min
    if t'_ab < t_ab then executes transfer: Remove i'_min from T_l
    and insert i'_min at position after j'_min in T_s. Update
    T_a, T_b.
until there is no improvement: t'_ab ≥ t_ab
```

6 Extensions

The previous sections provide the basic procedures to describe how STARS solves the 2RSRD problem. In this section, we present several extensions to STARS, such as

1. dealing with more than two-robot scenarios,
2. placing static sensors to enable constant area monitoring,
3. considering a nonuniform search and transmit interval and relay deploying time,
4. leveraging preexisting relays, and
5. leveraging extra carried relays to reduce dependence.

Extensions 1-3 are presented in this paper. The proof Theorem 3, extensions 4 and 5 are given in the online supplemental material. These extensions make STARS adaptive to more general conditions and varying user requirements.

6.1 Search with a Single or More than Two Robots

STARS trivially supports a single robot search since tour generation is simplified as PCATSP. In STARS, the solution to the first subproblem is irrelevant to the number of robots and is always valid. Therefore, STARS is extensible to search with more than two robots as long as PC2TSP is extensible to deal with more than two travelers.

Theorem 3. *Our solution to PC2TSP is extensible to deal with more than two travelers.*

The proof is presented in the online supplemental material.

6.2 Enable Constant Monitoring with Static Sensor Deployment

Search paths and relay deployment positions are computed offline before search starts. During the search process, additional sensors may be deployed at $SP_{susp} \subseteq \mathbb{SIP}$ to cover suspicious areas when they are identified during the search and transmission interval (STI), when human and/or computational analysis is conduct on the video streams at the base station. With the communication backbone to cover $\mathbb{SIP}$, the sensor deployed at any sensing position has a valid path back to the BS. Such sensor deployment significantly reduces the number of sensors compared to a full sensing coverage deployment. In addition, it adds on a constant monitoring on suspicious areas, compared to the single-time area scan and pass by the homogeneous mobile robots.

6.3 Consider a Nonuniform Search and Transmit Interval and Relay Deploying Time

We may define a nonuniform STI time $t_{STI}(i)$ for each $i \in \mathbb{SIP}$ and a relay deploying time $t_{rd}(j)$ for each $j \in \mathbb{RIP}$. The time only affects the cost matrix $c_{ij} \in \mathbb{C}$ in (10) for each edge ij. Besides the current value based on obstacle aware traverse path from i to j, any edge with a destination vertex of $j \in \mathbb{RIP}$ will add the value of $t_{rd}(j)$, $c_{ij} += t_{rd}(j)$, to consider the extra time of relay deployment. To take into account of the time for search and transmission interval, any edge with a starting vertex of $i \in \mathbb{SIP}$ will add the value of $t_{STI}(i)$, $c_{ij} += t_{STI}(i)$.

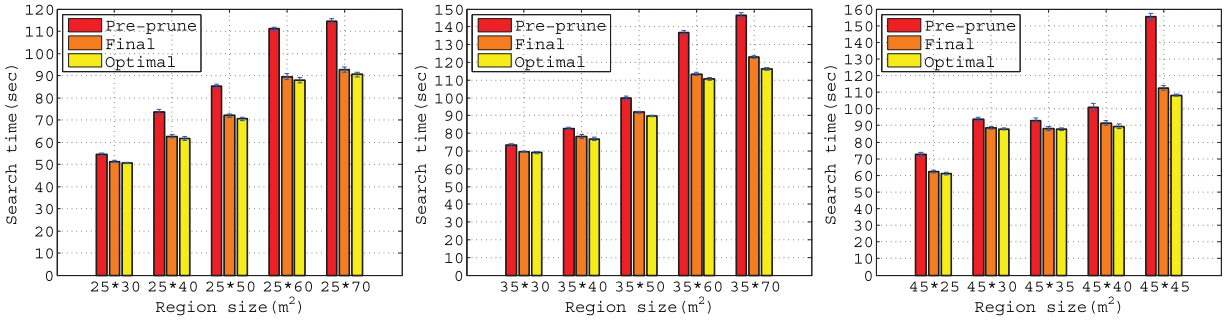
7 Performance Evaluation

We use a C++ simulation program to evaluate STARS search efficiency. PCATSP is computed in part by IBM ILOG CPLEX Optimizer V12.2. The region is represented by the grid-map model with $X \times Y$ cells. Cell edge length is 1 m, and the BS is located at the bottom center of the region. The robots travel at a constant speed of 1 m/s. $r_s = 10\,\mathrm{m}$ and $r_c = 20\,\mathrm{m}$. Obstacle cells are generated randomly each time with an obstacle ratio r_o from 5 to 25 percent as the parameter. *All* tests are run 200 times to obtain the average with 200 different obstacle distributions to form 200 different regions. When the obstacle ratio is high (i.e., 25 percent), many such obstacle distributions include convex shapes, gate shapes, or aggregated obstacles. The error bars in the figures show the 95 percent confidence interval.

First, we compare the search time with the optimal result in different shapes of regions with widths of 25 m, 35 m, and 45 m, respectively, as shown in Figs. 5a, 5b, and 5c ($r_o = 0.15$). The optimal result generator tries all possible tours with the same visiting positions and precedence constraint as STARS. Due to the high computation cost of the optimal result, we limit the region size where up to 18 visiting positions need to be traversed. The average difference of our heuristic from optimal is 1.94, 1.98, and 1.91 percent for the above three cases. Also, tour pruning and balance effectively reduce search time by 25 percent on average.

Fig. 6c shows the 30×50 grid cell case with varying densities of obstacles ($r_o = \{0.05, 0.1, 0.15, 0.2, 0.25\}$). As r_o increases from 5 to 25 percent, the optimal tour time increases from 88.6 to 90.8 s (2.4 percent). The difference of our heuristic from optimal increases slightly from 0.8 to 3.7 percent. We believe the near optimal results are in no small part contributed by 1) the optimal single robot tour by linear programming, and 2) the effective postpruning: reducing shared relays and vertex transfer.

We focus on narrow regions because a clear separation of two robots is then challenging. Also, the dependence between robots is likely to occur; the two robots tours are



(a) Extra-narrow region. (b) Narrow region. (c) Wide region.

Fig. 5. (a)-(c): Compare the search time with optimal under varying width regions (widths: 25-45 m).

likely to be mixed together rather than clearly separated. (Recall that the robots start searching from BS at the bottom center of the region.) In wider regions, it is possible to assign each robot a sub-area to search, converting the problem into a simpler single robot search problem.

Second, we show the search time trend with larger regions of 30*45, 40*60, 50*75, 60*90, 70*105 grid cells in Fig. 6a where optimal results are too slow to be obtained. Results show the two robot tours are well balanced. The differences between tours are only 6.8, 7.4, 4.1, 3.1, and 2.2 percent, respectively, as compared to the longer tour. Well-balanced tours do not guarantee near optimal performance but most optimal tours we observed are well balanced. Fig. 6b shows the number of visiting positions and relays in the above cases. The ratio of relays to visiting positions is 28.1, 36.5, 36.4, 36.0, and 35.1 percent, respectively, which shows that approximately one third of the positions need to be deployed relays.

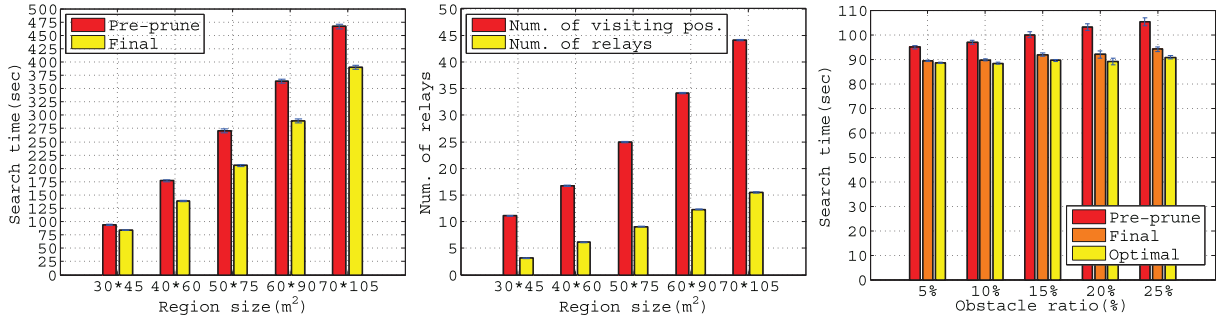
Third, we investigate on the impact of varying communication ranges and sensing ranges on the search time. Also, we evaluate the impact of varying STI and relay deploying time on search time. These results are discussed in the online supplemental material.

8 Related Work

A survey on multitraveling salesman (mTSP) is given in [10]. However, mTSP is to minimize the sum of the traveler tours rather than the maximum of them. Also, mTSP does not consider the precedence or the dependence between travelers. Balas and Simonetti [11] present a class of polynomially solvable (single) TSP with precedence constraints. However, their precedence constraint does not fit our application and is notably stricter than ours. In [11], a node i must be prior to *all* $i + k$ nodes given an initial order, whereas we only require i precedes a subset of nodes. Similarly unnecessary and stricter precedence constraint is imposed in the multiple traveling salesman problem with time windows (m-TSPTW) [12]. Besides, we do not have a predefined time window for each node.

There is little previous work on using online relay deployment to support remote sensing and surveillance with mobile robots and static sensors. Mobile robot search and exploration have been extensively studied without the consideration of online relay deployment [1], [13], [14], [15], [16], [17], [18], [19], [20]. Besides, hybrid static sensor and mobile robot systems have been studied in [21], [22], [23], [24], [25]. In [24], Kumar et al. deploy static sensors, RFID tags, and mobile robots to guide fire fighters in burning buildings. However, the static sensors are deployed ahead by humans. The sensors are not used as relays for real-time operation control. In [22], a single robot is used to assist the sensor network deployment and data collection. However, sensors are not used to transmit streams from the robot. In [26], Li et al. review algorithms on using mobile robots to serve wireless sensor networks. The focus of using robots is on deploying sensors for area coverage rather than deploying relays for communication.

One of the most relevant research work is presented in [25], where robots carry sensors as payload with online deployment. However, sensors are uniformly deployed.



(a) Search time in larger regions. (b) Num. of visiting pos. and relays. (c) Varied obstacle density.

Fig. 6. (a) Search time in larger regions. (b) Number of visiting positions and relays. (c) Varied obstacle density from 0 percent to 25 percent for 35*50 case.

They serve as the navigation guides rather than transmission relays. Similar ideas of using sensors to navigate robot movement are presented in [21] and [23].

9 Conclusion and Future Work

We first present a problem called precedence constrained two traveling salesman. A near-optimal heuristic to PC2TSP is presented with 2 percent average difference from optimal, which is in part contributed by optimal single tour computation, dependence handling, and effective pruning and balance. Leveraging PC2TSP, we propose a real-time search and monitoring scheme called STARS, which builds on a heterogeneous and cost-effective platform of mobile robots, static relays and sensors. Mobility of robots enables relay deployment on the fly to form a communication backbone at carefully selected locations. The backbone facilitates the robots to transmit streams to the base station for remote sensing and control. STARS significantly reduces cost by replacing mobile robots by static relays and enables constant area monitoring by sensor deployment only at specific suspicious areas rather than a full coverage.

Future work includes the following direction:

1. a real robot, relay, and sensor system implementation,
2. evaluating the impact of different relay sizes, relay deployment strategies, and carrying capacities on search time,
3. finding the optimal precedence constraint assignment or solving the problem with uncertain precedence constraints, and
4. the bandwidth aware relay placement and the probability based communication model.

Acknowledgments

This work was supported in part by US National Science Foundation (NSF) grant nos. OCI-0753362 and CNS-0721441. The preliminary version of this work appeared in the *Proceedings of the IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS) 2011* at Barcelona, Spain.

References

[1] Y. Pei, M. Mutka, and N. Xi, "Connectivity and Bandwidth Aware Real-Time Exploration in Mobile Robot Networks," *Wireless Comm. and Mobile Computing,* to be published.

[2] C. Moon, J. Kim, G. Choi, and Y. Seo, "An Efficient Genetic Algorithm for the Traveling Salesman Problem with Precedence Constraints," *European J. Operational Res.,* vol. 140, no. 3, pp. 606-617, 2002.

[3] X. Bai, S. Kumar, D. Xuan, Z. Yun, and T.H. Lai, "Deploying Wireless Sensors to Achieve Both Coverage and Connectivity," *Proc. ACM MobiHoc '06,* pp. 131-142.

[4] S. Guha and S. Khuller, "Approximation Algorithms for Connected Dominating Sets," *Algorithmica,* vol. 20, no. 4, pp. 374-387, 1998.

[5] Y. Pei and M. Mutka, "Joint Bandwidth-Aware Relay Placement and Routing in Heterogeneous Wireless Networks," *Proc. IEEE 17th Int'l Conf. Parallel and Distributed Systems (ICPADS '11),* pp. 1531-1536.

[6] D. Du and X. Hu, *Steiner Tree Problems in Computer Comm. Networks,* pp. 177-193, World Sci. Pub., 2008.

[7] S.C. Sarin, H.D. Sherali, and A. Bhootra, "New Tighter Polynomial Length Formulations for the Asymmetric Traveling Salesman Problem with and without Precedence Constraints," *Operations Research Letters,* vol. 33, no. 1, pp. 62-70, 2005.

[8] K. Hornik and M. Hahsler, "TSP-Infrastructure for the Traveling Salesperson Problem," *J. Statistical Software,* vol. 23, no. i02, pp. 1-18, 2007.

[9] N. Ascheuer, M. Jünger, and G. Reinelt, "A Branch & Cut Algorithm for the Asymmetric Traveling Salesman Problem with Precedence Constraints," *Computational Optimization and Applications,* vol. 17, pp. 61-84, 2000.

[10] T. Bektas, "The Multiple Traveling Salesman Problem: An Overview of Formulations and Solution Procedures," *Int'l J. Management Science,* vol. 34, no. 3, pp. 209-219, 2006.

[11] E. Balas and N. Simonetti, "Linear Time Dynamic-Programming Algorithms for New Classes of Restricted TSPs: A Computational Study," *Informs J. Computing,* vol. 13, no. 1, pp. 56-75, 2000.

[12] "The Multiple TSP with Time Windows: Vehicle Bounds Based on Precedence Graphs," *Operations Research Letters,* vol. 34, no. 1, pp. 111-120, 2006.

[13] A. Haumann, K. Listmann, and V. Willert, "Discoverage: A New Paradigm for multiRobot Exploration," *Proc. IEEE Int'l Conf. Robotics and Automation (ICRA '10),* pp. 929-934, 2010.

[14] P. Brass, F. Cabrera-Mora, A. Gasparri, and J. Xiao, "Multirobot Tree and Graph Exploration," *IEEE Trans. Robotics,* vol. 27, no. 4, pp. 707-717, Aug. 2011.

[15] J. Yuan, Y. Huang, T. Tao, and F. Sun, "A Cooperative Approach for Multi-Robot Area Exploration," *Proc. IEEE/RSJ Int'l Conf. Intelligent Robots and Systems (IROS '10),* pp. 1390-1395, 2010.

[16] Y. Pei, M. Mutka, and N. Xi, "Coordinated Multi-Robot Real-Time Exploration with Connectivity and Bandwidth Awareness," *Proc. IEEE Int'l Conf. Robotics and Automation (ICRA '10),* pp. 1531-1536, 2010.

[17] P. Mukhija, K. Krishna, and V. Krishna, "A Two Phase Recursive Tree Propagation Based Multi-Robotic Exploration Framework with Fixed Base Station Constraint," *Proc. IEEE/RSJ Int'l Conf. Intelligent Robots and Systems (IROS '10),* pp. 4806-4811, 2010.

[18] A. Marjovi, J. Nunes, L. Marques, and A. de Almeida, "Multi-Robot Exploration and Fire Searching," *Proc. IEEE/RSJ Int'l Conf. Intelligent Robots and Systems (IROS '09),* pp. 1929-1934, 2009.

[19] R. Rocha, F. Ferreira, and J. Dias, "Multi-Robot Complete Exploration Using Hill Climbing and Topological Recovery," *Proc. IEEE/RSJ Int'l Conf. Intelligent Robots and Systems (IROS '08),* pp. 1884-1889, 2008.

[20] J. Vazquez and C. Malcolm, "Distributed Multirobot Exploration Maintaining a Mobile Network," *Proc. IEEE Second Int'l Conf. Intelligent Systems,* vol. 3, pp. 113-118, June 2004.

[21] M. Batalin and G. Sukhatme, "The Design and Analysis of an Efficient Local Algorithm for Coverage and Exploration Based on Sensor Network Deployment," *IEEE Trans. Robotics,* vol. 23, no. 4, pp. 661-675, Aug. 2007.

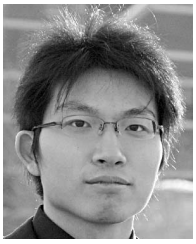
[22] Y. Wang and C.-H. Wu, "Robot-Assisted Sensor Network Deployment and Data Collection," *Proc. Int'l Symp. Computational Intelligence in Robotics and Automation (CIRA '07),* pp. 467-472, 2007.

[23] P. Corke, R. Peterson, and D. Rus, "Localization and Navigation Assisted by Networked Cooperating Sensors and Robots," *Int'l J. Robotics Res.,* vol. 24, pp. 771-786, 2005.

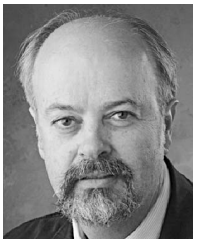
[24] V. Kumar, D. Rus, and S. Singh, "Robot and Sensor Networks for First Responders," *IEEE Pervasive Computing,* vol. 3, no. 4, pp. 24-33, Oct.-Dec. 2004.

[25] G. Fletcher, X. Li, A. Nayak, and I. Stojmenovic, "Back-Tracking Based Sensor Deployment by a Robot Team," *Proc. IEEE Seventh Ann. Comm. Soc. Conf. Sensor Mesh and Ad Hoc Comm. and Networks (SECON '10),* pp. 1-9, 2010.

[26] X. Li, R. Falcon, A. Nayak, and I. Stojmenovic, "Servicing Wireless Sensor Networks by Mobile Robots," *IEEE Comm. Magazine,* vol. 50, no. 7, pp. 147-154, July 2012.



Yuanteng Pei received the BEng degree in software engineering from Wuhan University, China, in 2007, and the PhD degree in computer science at Michigan State University in 2011. He has been an exchange student in computer science at City University of Hong Kong in 2005. He is currently a senior software engineer in the Cloud Engineering Group at eBay Inc. His research interests include wireless networking and mobile computing. Particularly, he is working on QoS support, remote sensing and control, motion and path planning on wireless mobile networks. He is a student member of the IEEE.



Matt W. Mutka received the BS degree in electrical engineering from the University of Missouri-Rolla, the MS degree in electrical engineering from Stanford University, and the PhD degree in computer science from the University of Wisconsin-Madison. He is on the faculty of the Department of Computer Science and Engineering, Michigan State University, East Lansing, Michigan, where he is currently a professor and department chairperson. He has been a visiting scholar at the University of Helsinki, Finland and a member of technical staff at Bell Laboratories in Denver, Colorado. His current research interests include mobile computing, wireless networking, and multimedia networking. He is a fellow of the IEEE.