# STARS: Static Relays for Multi-Robot Real-time Search and Monitoring

Yuanteng Pei and Matt W. Mutka

***Abstract*—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. Second, we show an application of PC2TSP and how our solution to PC2TSP can be applied to the application. By modeling in part by PC2TSP, we solve the problem of minimum time two-robot real-time search with relay deployment. We call the solution *STAtic Relay aided Search* (STARS), which identifies visiting positions by set cover and Steiner connected dominating set; assigns the precedence constraint by breadth-first search; and finally generates tours by PC2TSP. 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 with both narrow and wide regions show that our solution to PC2TSP achieves near-optimal performance with less than 2% average difference from optimal.**

***Keywords*-Multiple traveling salesman, precedence constraint, relay placement, coverage, motion planning, remote sensing.**

## I. Introduction

The traveling salesman problem (TSP) 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 $\mathbb{C} = \{1, 2, ..., n\}$ 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 traveling time between each pair of cities $i, j \in \mathbb{C}$, $c_{ij}$ and $c_{ji}$ may differ, 3) the precedence constraint that the arrival time plus waiting time at each consumer city is no earlier than the arrival time of all its supplier cities, to find two non-overlapping tours that allow waiting time at cities and both 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 multi-traveling 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 dependency between travelers.

Yuanteng Pei and Matt W. Mutka are with the Department of Computer Science and Engineering, Michigan State University, East Lansing, MI 48824, USA {peiyuant,mutka}@cse.msu.edu

This research was supported in part by NSF grants No. OCI-0753362 and CNS-0721441.

One interesting application of PC2TSP in mobile surveillance systems is 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 i) robots can efficiently search the area with minimum amount of time; ii) video streams from robots at any sensing position can be successfully transmitted back to the base station (BS). Fig. 1 illustrates 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 transmitted back 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 corresponding supplier city set is all of its supporting relay positions 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 *STAtic Relay aided Search* (STARS) to solve 2RSRD. STARS enables

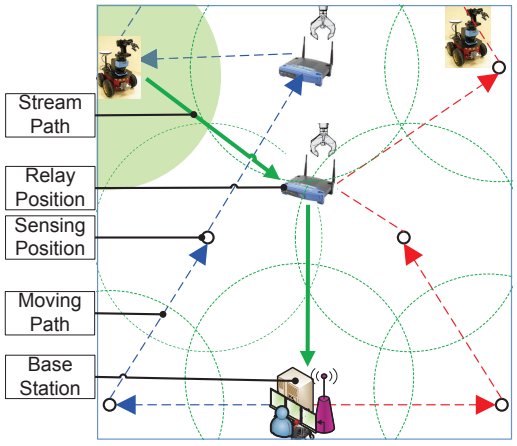



Fig. 1. Two robot search with static relay deployment.

remote robotic sensing and control. It has extensive applications such as reconnaissance, surveillance, search and rescue missions in dangerous areas such as the earthquake debris and radioactive environments. 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 VI-B.

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. The problem is motivated but has many challenges. We discuss these challenges in Section III-A.

### A. Main Contribution

- 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.
- To solve PC2TSP, we propose the tour validation and dependency checking scheme to detect deadlock and self-flipped cases that are invalid. It also checks the dependency 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 V. The rest of the paper proceeds as follows. Section II introduces the system model. In Section III, we formulate the problem and discuss the challenges. Section IV and V give the preprocessing and tour generation. Then, several extensions are discussed in Section VI. Performance evaluation is in Section VII and related work is summarized in Section VIII. Section IX concludes the paper and discusses the future work.

## II. System Model

*Environment Model*: We use a 2D occupancy map to model the environment as an unsearched rectangle region $\mathbb{R} = X \times Y$ with grid cells similar to that in [1]. The cell size equals to the robot size. A cell status include searched, unsearched, and obstructed. An obstacle cell cannot be visited by the robots nor can a relay be deployed on. 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. *Base station* **BS** 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 as 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 VI. The robots move at a constant speed where the acceleration and deceleration time is not considered. Each robot has traveling, communication, and sensing capability. It scans the environment using cameras and acoustic sensors with sensing range $\boldsymbol{r_s}$ and communicates with other nodes with communication range $\boldsymbol{r_c}$ by a 802.11 radio, where $r_c >= r_s$. We denote the communication to sensing range ratio as $\boldsymbol{\theta = r_c/r_s}$. Since only 2 robots are sending streams, we assume bandwidth is adequate in 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* **SP**, 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 non-obstacle 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*, e.g. 99%. $\mathbb{SP}$ is the set of SPs whose visit complete the search. $\mathbb{SP}$ includes the BS.

The robots deploy relays by visiting a *relay position* **RP** in the search tour. A relay is a non-mobile device with 802.11 for transmitting streams from a robot with same communication range $r_c$. A robot has a maximum carrying capacity of $n_r$ relays. We assume STI and relay deploying time are negligible.

SP(a) and SP(b) denote the vector of SPs robot a and b need to visit; RP(a) and RP(b) denote the vector of RPs a and b need to visit. $\boldsymbol{P_a}$, $\boldsymbol{P_b}$ denote the ordered vector of sensing and relay positions a and b traverse: $P_a = SP(a) \cup RP(a)$, $P_b = SP(b) \cup RP(b)$. $\boldsymbol{T_a}$, $\boldsymbol{T_b}$ 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 V=($v_0$, $v_1$,...$v_n$). $\boldsymbol{t_a}$ and $\boldsymbol{t_b}$ give the tour time in seconds, or the total time needs to traverse all positions in $P_a$ or $P_b$ including the possible waiting time. $t(i)$ 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. We do not model robots to come back to the BS immediately after the search is complete, because robots should remain in the frontier area for remote sensing.

*Discussions*: Due to its high complexity, the problem is

modeled as above and we focus on the tour generation. The model is easily extensible to consider a non-trivial STI and relay deploying time, which will be discussed in Section VI. In the future, we would like to tackle: 1) the bandwidth aware relay placement, 2) the probability based communication model, and 3) the online tour adjustment with inaccurate initial obstacle distribution. Note that the first two changes and most parameters in the model only affect the first subproblem in STARS and have no impact on the solution to the second subproblem of tour generation. For the third change, we may recompute the tour by updating the new data of obstacles as input.

## III. Problem formulation

Given the system model, 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:

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

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

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

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

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

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

Where $rp_i$ is a relay position in the path ($rp_i \in RP(a) \cup RP(b)$). (2) indicates the search is complete. (3)-(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 $t(i)$. (6) gives the carrying capacity constraint.

### A. 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:* Given an instance of a square region, we can exactly duplicate the region and put the duplicated copy on the right of the original one. With the BS on the bottom center of duplicated region, minimum $t_{ab}$ is achieved when both a and b traverse on its own half region the same way. This is because any traverse on the other half part will increase $t_{ab}$. Therefore, a polynomial time solution in 2RSRD will lead to a polynomial solution to a single *precedence constrained asymmetric traveling salesman problem* (**PCATSP**) on the positions. It contradicts to the fact that PCATSP is NP-hard [2], and this completes the proof. ■

There are other challenges to directly solve 2RSRD 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 dependency between both robots: One may have to wait for the other to deploy the supporting relays. The dependency 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 RPs are only placed selectively (such as at only S-CDS positions as in Section IV-B), 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 4 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 straight-forwardly solve 2RSRD to optimal has great complexity, 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 dependency occurs.

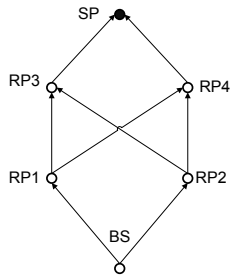



Fig. 2. Multiple choices of precedence constraints.

### B. STARS: STAtic Relay aided Search

Due to the problem's high complexity, we propose a high quality heuristic called **STARS**, or the *STAtic Relay aided Search*. 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 possible visiting position selections, precedence constraints and robot carrying capacities. With different user requirements, user's changing robot carrying capacity and intention of minimizing relays will not affect our tour computation. 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:

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

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

*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. We assume the user would like to save relays and with such an attempt the carrying capacity constraint is satisfied (Satisfying Eq.(6)). Other type of user's constraint on carrying capacity will be discussed in future work. Hence, the problem is: given $\mathbb{R}$ and $\mathbb{SP}$, how to place a minimum number of relay positions $\mathbb{RP}$ that satisfies 1) each SP is either in $\mathbb{RP}$ or a neighbor of RP; 2) $\mathbb{RP}$ includes the BS, and $\mathbb{RP}$ is connected:

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

*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 Eq.(3)-(5).

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

$$\text{Minimize } t_{ab} = \max\{t_a, t_b\}, \tag{9}$$
$$\text{Subject to } P_a \cup P_b = \mathbb{SP} \cup \mathbb{RP},$$
$$t(i) \leq t(j), \quad \forall p_{ij} = 1$$

*Rationale*: Problem 1a is formulated by minimum $|\mathbb{SP}|$ 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.

## IV. Preprocessing

### A. Identify Sensing Positions

Recall the problem formulation 1a in Section III-B. 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 non-obstacle cells become the universe. Each SP's sensing area is a set which 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. Alg. 1 describes the procedure.

**Algorithm 1**: Identify sensing positions for full region coverage.

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

### B. Identify Relay Positions

Recall the problem formulation 1b in Section III-B. We model the problem as the minimum *Steiner connected dominating set* (**S-CDS**) problem. CDS is a classical problem: 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 generation 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 non-obstacle cells in $\mathbb{R}$ as $V$. $\mathbb{RP}$ only needs to cover $\mathbb{SP}$, which is a subset of $V$. We require $\mathbb{RP}$ induce a connected subgraph because streams from any vertex in $\mathbb{SP}$ is sent to the BS.

**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 optimal DS finder.
4 **else**
5 Obtain DS by greedy set cover with maximum nodes.
*// 2. Apply Steiner tree approximation algorithm to connect DS.*
6 Call *Steinerized Minimum Spanning Tree* [5] to connect DS. The newly added nodes are $RP_2$
7 Return $\mathbb{RP} = DS \cup RP_2$.

The algorithm is similar to the Steiner-CDS algorithm in [4]. The 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 branch and bound based DS finder can compute up to 70 vertices to optimal in less than 1 minute, improving the S-CDS quality. **DS** is the overlapping set: DS$= \mathbb{SP} \cap \mathbb{RP}$. 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. Third, we always include the BS in DS because the BS belongs to the backbone: $BS \in \mathbb{RP}$.

### C. Assign the Precedence Constraint

Recall the problem formulation 1c in Section III-B. We first define notations, then present our solution and prove its validity. We define the *pure sensing positions* $\mathbb{SP}_{pure} = \mathbb{SP} \backslash \text{DS}$ ('\' means substraction), and *visiting positions* $\mathbb{VP} = \mathbb{SP} \cup \mathbb{RP}$.

Alg. 3 shows the procedure of assigning the precedence constraint by first assigning a *single* parentID for each node in $\mathbb{VP}$. 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 $\mathbb{VP}$. The tree has 1) the non-leaf backbone as $\mathbb{RP}$, 2) leaf nodes as $\mathbb{SP}_{pure}$, 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 Alg. 3 is valid.

*Proof:* Each node in $\mathbb{VP}$ has a single parentID. Because $\mathbb{SP} \subseteq \mathbb{VP}$, 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. ∎

## V. Tour Generation by PC2TSP

Recall the problem formulation 2 in Section III-B. 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 stand alone solution to PC2TSP with the changes as follows. 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 dependency checking. 3) Values in the cost matrix $c_{ij}$ and whether to have the conversion from Hamiltonian path depend on the application.

### A. Cluster Visiting Positions

We first cluster each nodes $i \in \mathbb{SP}$ 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. Although the initial $P_a$ and $P_b$ may be unbalanced, our tour pruning and balance in Section V-C work effectively in balancing the tours.

### B. 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 [6].

*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 [7]. 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:

$$\text{Minimize} \sum_{i=1}^{n} \sum_{j=1, j\neq i}^{n} c_{ij} x_{ij}, \tag{10}$$

$$\text{Subject to} \sum_{i=1, i\neq j}^{n} x_{ij} = 1, \quad \forall j = 1, ..., n,$$

$$\sum_{j=1, j\neq i}^{n} x_{ij} = 1, \quad \forall i = 1, ..., n,$$

$$p_{ij} \geq x_{ij}, \quad \forall i, j = 2, ..., n, \quad i \neq j,$$

$$p_{ij} + p_{ji} = 1, \quad \forall i, j = 2, ..., n, \quad i \neq j,$$

$$p_{ij} + p_{jk} + p_{ki} \leq 2, \quad \forall i, j, k = 2, ..., n, \quad i \neq j \neq k,$$

$$p_{ij} = 1, \quad \forall j = 2, ..., n, \quad \forall i \in SPC_j$$

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 while $p_{ij} = 1$ means position $i$ precedes $j$ not necessarily immediately in the tour. $SPC_j$ gives the subset of positions in $\{2, ..., n\}$ 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 [8].

### C. Prune and Balance Tours

*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 are self-reliant but may have

visited duplicate relay positions. Our goal is to reduce $t_{ab}$ by reducing duplicate RPs, considering the possibility and cost of introducing dependency with waiting time.

The pruning strategy has shown in Alg. 4. With $|SR|$ number of shared relays, we try all possible ways of allocating them to robot a and b by computing the powerset. Due to its 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 dependency checking is called in line 7 to check whether dependency occurs and to add waiting time, if it exists. Fig. 3(a) 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 $T_a, T_b$
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: $T_a = T_a \backslash j_{min}$, $T_b = T_b \backslash j'_{min}$.

**Algorithm 5**: Tour validation and dependency checking

1 **Input:** Local copies of any tours $T_a, T_b$.
2 **Output:** $t_{ab}$, updated $T_a, T_b$ with waiting time, isValid
3 isDeadlock=false, isValid=true.
4 **if** *isFlipped*$(T_a)$ || *isFlipped*$(T_b)$ **then** return $t_{ab}$=FLT_MAX, isValid=false.
5 Iterators $it_a, it_b$ at the beginning of $T_a, T_b$. $t_a = 0, t_b = 0$
6 **repeat**
7  **repeat**
8  $it_a$ 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.
9  **until** $it_a$ *is blocked when its parent is not visited.*
10  Same repeat procedure as above for $it_b$, which keeps on proceeding until it is blocked.
11  **if** *Both* $it_a, it_b$ *are blocked* **then** isDeadlock=$checkDeadlock(it_a, it_b)$.
12 **until** $it_a, it_b$ *reach the end of* $T_a, T_b$ || *isDeadlock*
13 **if** *isDeadlock* **then** return $t_{ab}$=FLT_MAX, isValid=false.

*Validate tours and check dependency*: 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 dependency.

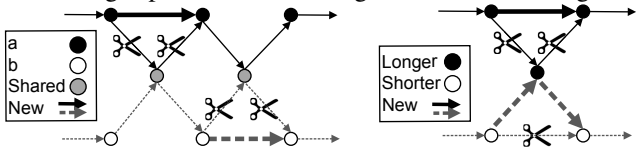
*Definitions*: A tour is *self-flipped* if and only if it has a position $i$ whose ancestors are visited after $i$. A self-flipped tour is invalid because by itself it violates the precedence constraint. A position $i$ is *blocked* when it has its parent not visited at time $t(i)$. $i$ has a missingID as its 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. 4(a) shows the robot arrives at a SP that requires a not yet deployed relay. Fig. 4(b) 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 Alg. 5.

**Algorithm 6**: Balance with vertex transfer

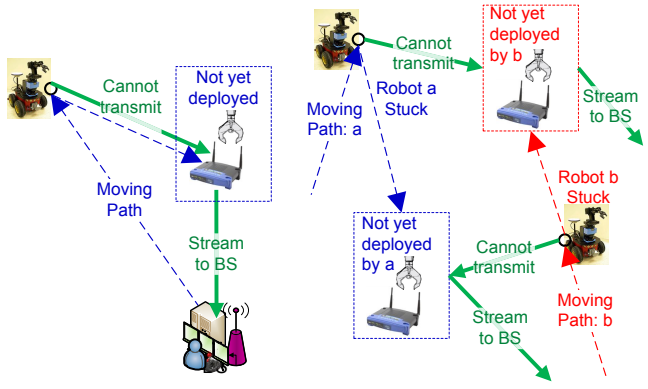
1 **Input:** $T_a, T_b$. **Output:** updated $T_a, T_b$
2 **repeat**
3  Mark the longer and shorter tours of $T_a, T_b$ as $T_l$, $T_s$.
4  **foreach** $i \in T_l$ **do**
5  **foreach** $j \in T_s, j \neq i$ **do**
6  Get new tours $T'_l \leftarrow (T_l$ with removed $i)$, $T'_s \leftarrow (T_s$ with inserted $i$ after position of $j)$.
7  Call Alg. 5 to validate tours and check dependency. Obtain the true time $t'_{ls}$ by adding wait time (if it exists) on $T'_l, T'_s$.
8  Get min $t'_{ab\_min}$ from all $t'_{ab}$, record corresponding $i'_{min}, j'_{min}$
9  **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 $T_a, T_b$.
10 **until** *there is no improvement:* $t'_{ab} \geq t_{ab}$

*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



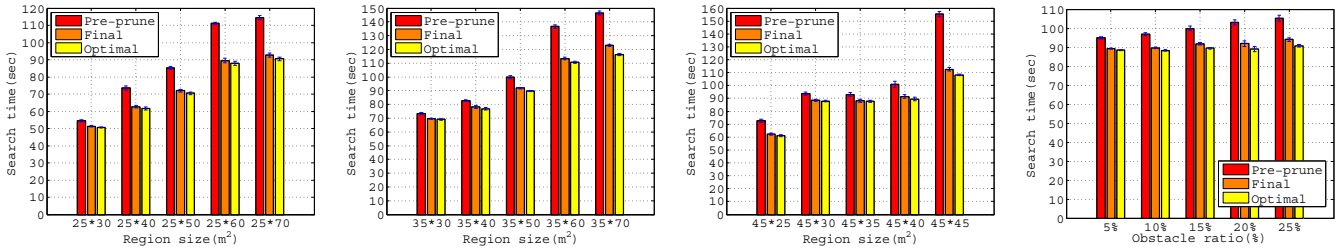

(a) Shared relay pruning. (b) Vertex transfer.

Fig. 3. Shared relay pruning and vertex transfer.




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

Fig. 4. Invalid cases.




(a) Extra-narrow region. (b) Narrow region. (c) Wide region. (d) Varied obstacle density.

Fig. 5. (a)-(c): Compare the search time with optimal under varying width regions (widths: 25-45m). (d) Varied obstacle density from 0% to 25%.

the shorter one. We greedily choose the best pair of removal and insertion position until no further reduction is found. The procedure is in Alg. 6 with illustration in Fig. 3(b).

## VI. Extensions

### A. 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.

*Proof:* We first illustrate the changes with $n$=3 robots of $R_0, R_1, R_2$ and then extend it with $n > 3$.

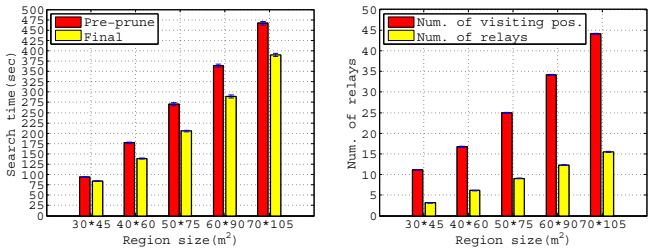
The sensing positions are clustered by $3-1=2$ vertical lines $X = \mathbb{R}_x/3$, $X = 2 \cdot \mathbb{R}_x/3$ to evenly partition the region. Pruning is first conducted the same as before to relays only shared by two robots $SR_{01}, SR_{02}, SR_{12}$. The shared relays of all 3 robots $SR_{012}$ , if they exist, are then partitioned and evaluated by a recursive powerset computation. In tour validation, deadlock checking needs to consider the additional cases with 3-robot interdependency. Finally, vertex transfer between each robot pair (0,1),(0,2),(1,2) is executed until the search time cannot be reduced any further.

Cases with $n > 3$ are similarly extended. $\mathbb{SP}$ are clustered by $n-1$ lines. The shared relays are evaluated with $\sum_{i=2}^{n} \binom{n}{i} \cdot 2^{\lambda}$ different cases, where $\lambda$ is the number of shared relays between that set of robots. Up to $n$ interdependencies are checked for deadlocks. Besides, vertex transfer between $\binom{n}{2}$ pairs of robots is evaluated. ■

When $n > 5$, the procedure of shared relay pruning may be simplified due to high computation cost. However, with the goal of replacing robots by relays and the strategy of partitioning areas, we believe $n \leq 5$ is sufficient in most cases.

### B. 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 $SP_{susp} \subseteq \mathbb{SP}$ to cover suspicious areas when they are identified after analysis on the video streams at the BS. With the communication backbone to cover $\mathbb{SP}$, the sensor deployed at any sensing position has a valid path back to the BS. Such sensor deployment significantly reduces the sensor size compared to a full sensing coverage deployment.




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

Fig. 6. (a) Search time in larger regions. (b) Num. of visiting pos. and relays.

### C. Consider a non-uniform search and transmit interval and relay deploying time

We may define a non-uniform STI time $t_{STI}(i)$ for each $i \in \mathbb{SP}$ and a relay deploying time $t_{rd}(j)$ for each $j \in \mathbb{RP}$. The time only affects the cost matrix $c_{ij} \in \mathbb{C}$ in Eq. 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 $j \in \mathbb{RP}$ will add the value of $t_{rd}(j)$, $c_{ij} += t_{rd}(j)$. Any edge with a starting vertex of $i \in \mathbb{SP}$ will add the value of $t_{STI}(i)$, $c_{ij} += t_{STI}(i)$.

## VII. Performance Evaluation

We use a C++ simulation program to evaluate STARS search efficiency. PCATSP is computed in part by CPLEX 12.2. The region is represented by the grid-map model with $X \times Y$ cells. Cell edge length is 1 meter, and the BS is located at the bottom center of the region. The robots travel at a constant speed of 1m/sec. $r_s$=10m and $r_c$=20m. Obstacle cells are generated randomly each time with an obstacle ratio $r_o$ as the parameter. All tests are run 200 times to obtain the average. The error bars show 95% confidence interval.

First, we compare the search time with the optimal result in different shapes of regions with widths of 25m, 35m, and 45m respectively as shown in Figures 5(a)-5(c) ($r_o$=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 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% for the above 3 cases. Also, tour pruning and balance effectively reduce search time by 25% on average.

Fig. 5(d) 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%, the optimal tour time increases from 88.6 to 90.8 seconds (2.4%). The difference of our heuristic from optimal increases slightly from 0.8% to 3.7%.

We focus on narrow regions because a clear separation of two robots is then challenging, and dependency between robots is likely to occur with more tour mixture. 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. 6(a) 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% 2.2% 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.6(b) 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% respectively, which shows that approximately one third of the positions need to be deployed relays.

## VIII. Related Work

A survey on multi-traveling salesman (mTSP) is in [9]. However, mTSP is to minimize the sum of the tours rather than the maximum of them. Also, mTSP does not consider precedence or dependency. A class of polynomially solvable (single) TSP with precedence constraints is given in [10]. However, their precedence constraint does not fit our application and is notably stricter than ours where a node $i$ must be prior to *all* $i + k$ nodes, whereas we only require $i$ precedes a subset of nodes. Similarly unnecessary and stricter precedence constraint is imposed in the problem of mTSP with time windows (m-TSPTW) [11]. 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 are in [12–17]. Besides, hybrid static sensor and mobile robot systems have been studied in [18–22]. In [21], authors deploy static sensors, RF tags, and mobile robots to guide fire fighters in burning buildings. However, the static sensors are deployed ahead by humans. Sensors are not used as relays for real-time operation control. In [19], 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.

One of the most relevant work is in [22], where robots carry sensors as payload with online deployment. However, sensors are uniformly deployed. They serve as navigation guides rather than transmission relays. Similar ideas of using sensors to navigate robot movement are presented in [18,20].

## IX. Conclusion and Future Work

We first present a problem called precedence constrained two traveling salesman (PC2TSP). A near-optimal heuristic to PC2TSP is presented with 1.94% average difference from optimal, which is in part contributed by optimal single tour computation, dependency 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: 1) 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) those discussed in Section II,III.

## References


[1] Y. Pei, M. Mutka, and N. Xi, "Coordinated multi-robot real-time exploration with connectivity and bandwidth awareness," in *ICRA 2010*.

[2] C. Moon, J. Kim, G. Choi, and Y. Seo, "An efficient genetic algorithm for the traveling salesman problem with precedence constraints," *European J. of Operational Res.*, vol. 140, no. 3.

[3] X. Bai, S. Kumar, D. Xuan, Z. Yun, and T. H. Lai, "Deploying wireless sensors to achieve both coverage and connectivity," in *MobiHoc 2006*.

[4] S. Guha and S. Khuller, "Approximation algorithms for connected dominating sets," *Algorithmica*, vol. 20, no. 4, pp. 374–387, 1998.

[5] D. Du and X. Hu, *Steiner Tree Problems In Computer Communication Networks*. World Scientific Publishing Co., 2008, pp. 177–193.

[6] 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.

[7] K. Hornik and M. Hahsler, "TSP–infrastructure for the traveling salesperson problem," *J. of Statistical Software*, vol. 23, no. i02, 2007.

[8] N. Ascheuer, M. Jünger, and G. Reinelt, "A branch & cut algorithm for the asymmetric traveling salesman problem with precedence constraints," *Comput. Optim. Appl.*, vol. 17, 2000.

[9] T. Bektas, "The multiple traveling salesman problem: an overview of formulations and solution procedures," *Omega, Int. J. Management Sci.*, vol. 34, no. 3, 2006.

[10] E. Balas and N. Simonetti, "Linear time dynamic-programming algorithms for new classes of restricted TSPs: A computational study," *INFORMS J. on Computing*, vol. 13, no. 1, 2000.

[11] "The multiple TSP with time windows: vehicle bounds based on precedence graphs," *Operations Research Letters*, vol. 34, no. 1, 2006.

[12] A. Haumann, K. Listmann, and V. Willert, "Discoverage: A new paradigm for multi-robot exploration," in *ICRA 2010*.

[13] P. Brass, F. Cabrera-Mora, A. Gasparri, and J. Xiao, "Multirobot tree and graph exploration," *Robotics, IEEE Transactions on*, no. 99, 2011.

[14] J. Yuan, Y. Huang, T. Tao, and F. Sun, "A cooperative approach for multi-robot area exploration," in *IROS 2010*.

[15] Y. Pei, M. Mutka, and N. Xi, "Connectivity and bandwidth aware real-time exploration in mobile robot networks," in *Wiley Wireless Communications and Mobile Computing Journal, WCMC, (in press)*.

[16] P. Mukhija, K. Krishna, and V. Krishna, "A two phase recursive tree propagation based multi-robotic exploration framework with fixed base station constraint," in *IROS 2010*.

[17] A. Marjovi, J. Nunes, L. Marques, and A. de Almeida, "Multi-robot exploration and fire searching," in *IROS 2009*.

[18] M. Batalin and G. Sukhatme, "The design and analysis of an efficient local algorithm for coverage and exploration based on sensor network deployment," *Robotics, IEEE Transactions on*, vol. 23, no. 4, 2007.

[19] Y. Wang and C.-H. Wu, "Robot-assisted sensor network deployment and data collection," in *CIRA 2007*.

[20] P. Corke, R. Peterson, and D. Rus, "Localization and navigation assisted by networked cooperating sensors and robots," *Int. J. Rob. Res.*, 2005.

[21] V. Kumar, D. Rus, and S. Singh, "Robot and sensor networks for first responders," *Pervasive Computing, IEEE*, vol. 3, no. 4, 2004.

[22] G. Fletcher, X. Li, A. Nayak, and I. Stojmenovic, "Back-tracking based sensor deployment by a robot team," in *IEEE SECON 2010*.