Report Number: WUCSE-2006-54

2006-01-01

Mixed-Integer Linear Programming Solution to Multi-Robot Task Allocation Problem

Nuzhet Atay and Burchan Bayazit

Multi-robot systems require efficient and accurate planning in order to perform mission-critical tasks. This paper introduces a mixed-integer linear programming solution to coordinate multiple heterogenenous robots for detecting and controlling multiple regions of interest in an unknown environment. The objective function contains four basic requirements of a multi-robot system serving this purpose: control regions of interest, provide communication between robots, control maximum area and detect regions of interest. Our solution defines optimum locations of robots in order to maximize the objective function while efficiently satisfying some constraints such as avoiding obstacles and staying within the speed capabilities of the... **Read complete abstract on page 2.**

Mixed-Integer Linear Programming Solution to Multi-Robot Task Allocation Problem

Nuzhet Atay and Burchan Bayazit

Complete Abstract:

Multi-robot systems require efficient and accurate planning in order to perform mission-critical tasks. This paper introduces a mixed-integer linear programming solution to coordinate multiple heterogenenous robots for detecting and controlling multiple regions of interest in an unknown environment. The objective function contains four basic requirements of a multi-robot system serving this purpose: control regions of interest, provide communication between robots, control maximum area and detect regions of interest. Our solution defines optimum locations of robots in order to maximize the objective function while efficiently satisfying some constraints such as avoiding obstacles and staying within the speed capabilities of the robots. We implemented and tested our approach under realistic scenarios. We showed various extensions to objective function and constraints to show the flexibility of mixed-integer linear programming formulation.

2006-54

Mixed-Integer Linear Programming Solution to Multi-Robot Task Allocation Problem

Authors: Nuzhet Atay, Burchan Bayazit

Corresponding Author: atay@cse.wustl.edu

Abstract: Multi-robot systems require efficient and accurate planning in order to perform mission-critical tasks. This paper introduces a mixed-integer linear programming solution to coordinate multiple heterogenenous robots for detecting and controlling multiple regions of interest in an unknown environment. The objective function contains four basic requirements of a multi-robot system serving this purpose: control regions of interest, provide communication between robots, control maximum area and detect regions of interest. Our solution defines optimum locations of robots in order to maximize the objective function while efficiently satisfying some constraints such as avoiding obstacles and staying within the speed capabilities of the robots. We implemented and tested our approach under realistic scenarios. We showed various extensions to objective function and constraints to show the flexibility of mixed-integer linear programming formulation.

Type of Report: Other

Mixed-Integer Linear Programming Solution to Multi-Robot Task Allocation Problem

Nuzhet Atay
Department of Computer Science and Engineering
Washington University in St. Louis
Email: atay@cse.wustl.edu

Burchan Bayazit
Department of Computer Science and Engineering
Washington University in St. Louis
Email: bayazit@cse.wustl.edu

Abstract— **Multi-robot systems require efficient and accurate planning in order to perform mission-critical tasks. This paper introduces a mixed-integer linear programming solution to coordinate multiple heterogenous robots for detecting and controlling multiple regions of interest in an unknown environment. The objective function contains four basic requirements of a multi-robot system serving this purpose:** *control regions of interest*, *provide communication between robots*, *control maximum area* **and** *detect regions of interest*. **Our solution defines optimum locations of robots in order to maximize the objective function while efficiently satisfying some constraints such as avoiding obstacles and staying within the speed capabilities of the robots. We implemented and tested our approach under realistic scenarios. We showed various extensions to objective function and constraints to show the flexibility of mixed-integer linear programming formulation.**

I. Introduction

Several real life scenarios, such as fire fighting, search and rescue, surveillance, etc., need multi-robot coordination and task allocation. Such scenarios generally include distinct regions of interests that require the attention of some robots. If the locations of these regions are not known, the robots also need to explore the environment to find them. For example, in a forest fire, the fire can start at different locations and fire fighters may need to detect the fire first and then extinguish it. The size of the burning area may require more fire teams in some locations. Similarly, in a surveillance scenario, there can be multiple intruders in different locations and each intruder may show a different behavior. In order to coordinate the task at hand, it may also be helpful to keep individual robots in contact with each other so that the information is shared by as many robots as possible.

In this paper, we propose a solution to the problem of detecting and controlling multiple regions of interest in an unknown environment using multiple robots. In our system, we assume a bounded environment that is to be controlled by a group of robots. In this environment, there are regions of interest which need to be tracked. These regions are dynamic, i.e. they can appear at any point, anytime and can move, spread or disappear. Each region may require more than one robot to track and control. Robots do not have initial information about the environment, and the environment is only partially-observable by the robots. Each robot has wireless communication capability, but its range is not uniform. Two robots can communicate between each other only if both of them are in the communication range of each other. Robots can be mobile and have different speed limits. The robots are equipped with the sensors to identify the obstacles and the regions of interest if they are within the robots' sensing range. Ranges of these sensors are not necessarily uniform. The environment can have static or dynamic obstacles, so robots need to avoid them to be able to perform their task.

Our solution to the task allocation problem of heterogeneous robots utilizes mixed integer linear programing with an optimization criterion and some resource constraints. Optimization criterion we are interested in is: (i) *covering all regions of interest*, (ii) *providing communication between as many robots as possible*, (iii) *controlling maximum total surface by all the robots*, (iv) *exploring new regions*. Our objective is to maximize these items while satisfying the constraints such as avoiding the obstacles or moving within the speed capabilities of individual robots. Additional constraints we are considering are the communication between two robots (which exists only if either two robots are in the communication range of each other or there is a route between them through other robots satisfying the communication constraints) and, the sensing of the obstacles and regions of interest when they are within the robot's sensor range. However, our approach is general enough to easily adapt to additional constraints and objectives, making it customizable for various problems. In order to validate our approach, we have formulated and implemented a linear program, and tested it under realistic scenarios. We have shown sample modifications and extensions to formulate additional constraints or objectives. Our main contribution is a customizable multi-robot task allocation solver which can find global optimal solution under the given constraints. One advantage of our method is that we are considering the future predictions so that our task allocation would give the optimal solution within a predifined time period. Our approach also proposes an efficient way of checking obstacle collision using linear programming.

The rest of the paper is organized as follows. The next section gives a summary of the related research and brief comparison to our approach when it is applicable. Section III gives problem definition and describes the basic variables. Section IV describes our solution. Section V presents experimental results. We discuss extensions in Section VI and Section VII concludes our paper.

II. Related Work

Multi-robot task allocation has been studied extensively because of the importance of application areas. One quite popular approach to this problem is utilizing negotiation or auction based mechanisms. In this approach, each distributed agent computes a cost for completing a task, and broadcasts the bid for that task. Auctioneer agent decides the best available bid, and winning bidder attempts to perform this task. Following the contract-net protocol [1], several variations of this method has been proposed [2]–[6]. Another important approach is using behaviour based architecture. ALLIANCE [7] is a behavior-based architecture where robots use motivational behaviors such as robot impatience and robot acquiescence. These behaviors motivate robots to perform tasks that cannot be done by other robots, and give up the tasks they cannot perform efficiently. BLE [8] is another behavior-based architecture which uses continous monitoring of tasks among robots and best fit robot is assigned to each task. A detailed analysis and comparison of these methods can be found at [9], [10]. These methods propose distributed algorithms where resource allocation is an approximation to the global optimum. Although they were shown to be successful in practice, they do not guarantee global optimum even if communication between all robots is provided. Our solution, however, guarantees global optimum if all robots can communicate between each other.

Task allocation problem is also studied in the context of cooperation of Unmanned Aerial Vehicles (UAVs). Several methods are proposed for search and attack missions of UAVs [11]–[19]. Our method is similar to the methods proposed in [12], [13], [16], [19], since these methods are also using mixed-integer linear programming task allocation. However, in these papers, the problem is defined as minimizing mission completion time while UAVs visiting predetermined waypoints and avoiding no-fly zones. The solution to this problem is formulated as finding all possible combinations of task allocations, and choosing the best combination. This definition of task allocation is actually quite different than our problem definition. Our aim is to explore environment, find regions of interest, and assign tasks optimally obeying the constraints imposed at that moment. In other words, we are finding a solution in real-time, instead of finding an initial plan and executing it.

III. Problem Definition

In our problem definition, there are regions of interest we want robots to explore and cover. In the rest of the paper, we will call these regions "targets". Since larger areas can be represented with multiple points, without loss of generality, we assume targets are represented as points in planar space. A target is assumed to be covered if there are enough robots that have the target in their sensing range. The number of robots required to cover a target varies for each target. We assume the future locations of known targets after a time period can be predicted. Our primary purpose is to find locations of robots in order to cover as many targets as possible, using the estimated locations of targets. While covering all targets, it is also desirable to provide communication between as many robots as possible because this increases robots' information about the environment and targets, so leads to a better solution. At a given time, robots need to cover as much area as possible besides covering targets to increase the chances of detecting other undiscovered targets. Similarly, in order to discover new targets and avoid waiting at the same location when no targets are being tracked, the robots are expected to explore new regions.

We define the state of the system as current locations of the targets, number of robots needed to cover a target, current positions of the robots, positions of obstacles, previously explored regions, robot speed, communication range and sensing range. The output of our linear program is the optimal locations of the robots for the next state of the system after a brief period of time. Please note that, we assume we can predict the location of the targets at the next step. There are approaches for motion prediction that can be used for this purpose [20].

A. Variables

Our planning is for the next state, so all variables take values according to next state unless stated otherwise. Variables in our linear program formulation can be listed as the following (note that "[r]" represents a real, "[i]" represents an integer, and "[b]" represents a binary variable):

- (r_i^x, r_i^y) [r]: final position of robot i.
- $distance_{ij}^{RT}$ [r]: distance between robot i and target j.
- $distance_{ij}^{RR}$ [r]: distance between robots i and j.
- $movement_i$ [r]: distance between between initial (current state) and final (next state) positions of robot i.
- $coverage_j$ [b]: indicates whether a target is covered.
- $communication_{ij}$ [b]: indicates whether a communication link between robots i and j exists.
- $area_{ij}$ [b]: specifies whether sensing ranges of robots i and j overlap.
- $exploration_{ij}$ [b]: indicates whether the robot i will be inside the explored region j.
- $proximity_{ij}$ [b]: shows if robot i can sense target j.
- bh_{ij}^x, bh_{ij}^y, bl_{ij}^x, bl_{ij}^y [b]: represents whether there is a possible path between initial (current state) and final (next state) positions of the robot i not blocked by obstacle j.
- mp_i^x, mn_i^x, mp_i^y, mn_i^y [r]: used in finding an alternative path of robot i to avoid obstacles in the straightforward path.
- ieh_{ij}^x, ieh_{ij}^y, iel_{ij}^x, iel_{ij}^y [b]: used for finding position of the robot i with respect to explored region j.

Besides these, there are constants used in specifying objective function and contraints:

- (ir_i^x, ir_i^y) [r]: initial position of robot i at the current state.
- (t_j^x, t_j^y) [r]: estimated position of target j at the next state.
- $coverageRequirement_j$ [i]: number of robots needed to cover target j.
- $sensingRange_i$ [r]: range of sensors on robot i.
- $robotSpeed_i$ [r]: speed of robot i, defined as the number of unit steps it can go by moving in x-axis or y-axis at each step.
- $timeStep$ [r]: time range during which this planning takes place.

- $commRange_i$ [r]: communication range of robot i.
- (oh_j^x, oh_j^y), (ol_j^x, ol_j^y) [r]: upper right and lower left corners of obstacle j, respectively.
- (eh_j^x, eh_j^y), (el_j^x, el_j^y) [r]: upper right and lower left corners of explored region j, respectively.

IV. Mixed Integer Linear Programming for Task Allocation

We use linear programming to find the best placement of the robots after a defined time period. This program runs for all the robots that are in the communication range of each other. If there are multiple groups of robots that cannot communicate with each other, each group will have its own task allocation based on its world view. If two groups merge, they can share their knowledge. The program runs periodically to find the best placements for each robot. It also runs if a new event happens, such as the discovery of an obstacle or a target. The linear program should satisfy some constraints: (i) an evaluated location is not acceptable if the robot cannot reach there either because of its speed limits or because of an obstacle, (ii) two robots cannot communicate if one of them is outside the communication range of the other, (iii) an obstacle or target is detectable only if it is within the sensing range of the robot. Our goal is then to optimize the number of targets tracked, the number of robots that can communicate with each other, the area of the environment covered by the robot sensors, and the area of the environment that was explored. In the next subsections, we will first define the basic functions and constraints. Then we will discuss different objective functions. Finally we show our overall optimization criterion and we will discuss the complexity.

A. Basic Functions

Distance between robots and targets: [1]

$$distance_{ij}^{RT} = |r_i^x - t_j^x| + |r_i^y - t_j^y| \quad (1)$$

for each target j and each robot i.

Distance between robots: [1]

$$distance_{ij}^{RR} = |r_i^x - r_j^x| + |r_i^y - r_j^y| \quad (2)$$

for each robot pair i and j.

Movement of robots: (i.e., the distance between initial and goal positions) [1]

$$movement_i = |r_i^x - ir_j^x| + |r_i^y - ir_j^y| \quad (3)$$

for each robot i.

B. Basic Constraints

Robots have limited speed, so their final position should not be beyond their reaching limit:

$$movement_i \leq timeStep * robotSpeed_i \quad (4)$$

for each robot i. Note that, as we will see next, we also consider detours if there is an obstacle on the direct path.

In our system, we assume there are only rectangular shaped obstacles for the sake of simplicity of defining linear equations. However, more general shaped obstacles

[1]In order to satify the linear properties, we use manhattan distance. See the appendix for linear modelling of absolute value function.

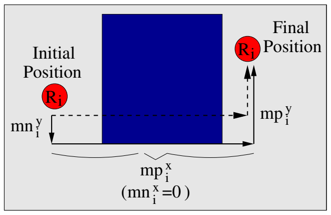

Fig. 2. An example obstacle avoidance for robot r_i. Values of mp^x, mp^y, mn^x and mn^y are arranged to verify that an alternative and feasible path exists. Dashed line shows straightforward path, straight line shows updated path.

can be represented as rectangular meshes. When considering obstacles, we are not finding a path to avoid them, but finding whether or not it is possible to avoid them with the robot speed and timestep as the constraints. As it is mentioned before, output of the linear program is the final positions of robots. When computing these positions, we are also computing the manhattan path length of the robot to avoid an obstacle if the obstacle is in the robot's way. If this avoidance is possible with the particular robot speed and timestep, we are condering the final position of the robot as a feasible configuration. Otherwise, that configuration is eliminated. However, we are not finding the particular path that leads robot to the final position, that step is left to a local planner running on the robot. Obstacle detection is much more efficient this way than finding the exact path using linear program. Finding exact path requires finding intermediate states of the system at a fine resolution which increases complexity drastically.

$$\begin{aligned}
& bh_{ij}^x = 0,\ where\ ir_i^x + mp_i^x \geq oh_j^x \quad (5)\\
& bh_{ij}^y = 0,\ where\ ir_i^y + mp_i^y \geq oh_j^y\\
& bl_{ij}^x = 0,\ where\ ir_i^x - mn_i^x \leq ol_j^x\\
& bl_{ij}^y = 0,\ where\ ir_i^y - mn_i^y \leq ol_j^y\\
& mp_i^x + mp_i^y + mn_i^x + mn_i^y \leq timeStep * robotSpeed_i\\
& r_i^x - ir_i^x = mp_i^x - mn_i^x\\
& r_i^y - ir_i^y = mp_i^y - mn_i^y\\
& Obstacle\ is\ not\ avoidable\ if;\\
& bh_{ij}^x = 1\ and\ bl_{ij}^x = 1\ and\ bh_{ij}^y = 0\ and\ bl_{ij}^y = 0\\
& bh_{ij}^y = 1\ and\ bl_{ij}^y = 1\ and\ bh_{ij}^x = 0\ and\ bl_{ij}^x = 0
\end{aligned}$$

for each robot i and obstacle j. In the formulation above, first, some alternative initial and goal positions are found. The first four constraints guarantee that a path between alternative positions would avoid the obstacle. Next, the feasibility of such an alternative path is evaluated (i.e., reaching the alternative initial position from the original initial position, reaching the alternative goal position using manhattan path and reaching the original goal position from the alternative goal position must be within the speed limits of the robot). The variables mp^x, mp^y, mn^x and mn^y represent the offsets that will generate alternative placements of the initial and final positions. Their usage is represented in Fig. 2. In this figure, offsetting is required in $y-axis$, which is done by arranging values of mp^y and mn^y. There is no change made in $x-axis$, so mp^x shows the correct

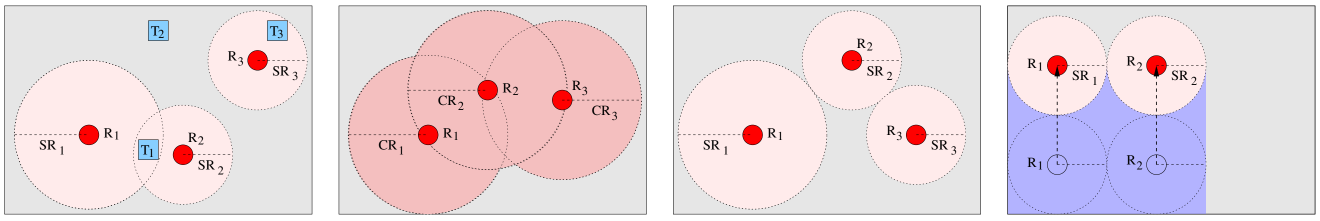

Fig. 1. SR stands for sensing range, and CR stands for communication range (a) A target is covered when it is in sensing range of some robots, where number of robots is determined according to the requirements of the target. Robots R_1 and R_2 cover T_1, while R_3 covers T_3. T_2 is not covered. (b) Two robots can communicate if both robots are in communication range of each other. R_2 can communicate with R_1 and R_3, and works as a hub between R_1 and R_3 which cannot communicate directly. (c) Maximum area coverage is obtained if sensing range of robots do not overlap. In the figure, sensing regions of robots barely touch each other (d) Robots mark regions they explored before, and move towards unexplored regions. R_1 and R_2 move upward toward unexplored region after marking dark (blue) region as explored

distance in $x-axis$ while mn^x is 0. Variables bh^x_{ij}, bl^x_{ij} and bh^y_{ij}, bl^y_{ij} indicate location of obstacle j with respect to the offset initial and final positions of robot i. If both bh^x_{ij} and bl^x_{ij} are 1, this means that obstacle j is between the offset initial and final positions of robot i on $x-axis$. In that case, if both bh^y_{ij} and bl^y_{ij} are 0, then both the offset initial and final positions of robot i are inside obstacle j on $y-axis$. So, there is no manhattan path connecting initial and final positions. The same is true with x and y axes interchanged.

C. Target Coverage

A target can be considered covered only if the number of robots following it is greater than or equal to its coverage requirement:

$$coverage_j = 1,\ where \sum_{i=1}^{n} proximity_{ij} \geq coverageRequirement_j \tag{6}$$

for each target j. [2]

A robot can sense and control a target only if its sensing range is greater than or equal to the distance between itself and the target:

$$proximity_{ij} = 1,\ where\ distance^{RT}_{ij} \leq sensingRange_i \tag{7}$$

for each target j and each robot i.

A sample organization of the robots and targets is shown in Fig. 1(a). *R1* and *R2* are covering target *T1* and *R3* is covering *T3* while *T2* is not covered by any of the robots.

D. Communication

Each robot has a communication range. A robot can have a duplex communication link to another robot only if each robot is in the sensing range of the other one:

$$\begin{aligned} &communication_{ij} = 1, \\ &where\ distance^{RR}_{ij} \leq commRange_i \\ &and\ distance^{RR}_{ij} \leq commRange_j \end{aligned} \tag{8}$$

for each robot pair i and j.

[2]Please see the appendix for the proof that our optimization criterion results in continuous target coverage of all targets, if this optimization has highest priority.

However, robots can communicate between each other with the help of other robots. So, if two robots cannot directly communicate with each other, but they share common robot both of which can communicate, we assume that they can communicate. In other words, transitive links are allowed in the system. It should be noted that this condition implies communication between robots with the help of multiple intermediate robots, i.e. one or more robots can participate in a transitive link between two robots.

$$\begin{aligned} &communication_{ij} = 1, \\ &where\ communication_{ik} + communication_{kj} = 2 \end{aligned} \tag{9}$$

for each robot i, j and k.

A communication pattern of the robots is shown in Fig. 1(b). *R2* can communicate with both *R1* and *R3*. *R1* and *R3* do not have a direct communication link, but they can communicate with the help of *R2*.

E. Area Coverage

Robots have limited and constant sensing range, so the only way to maximize area coverage is by preventing the overlap of sensing ranges of robots.

$$\begin{aligned} &area_{ij} = 1, \\ &where\ distance^{RR}_{ij} \geq sensingRange_i + sensingRange_j \end{aligned} \tag{10}$$

for each robot pair i and j.

An ideal area coverage for the robots is represented in Fig. 1(c), where robots have no overlapping sensing range.

F. Exploration

In order to explore the environment, robots need to know places they have visited recently. We store this information as rectangular regions defining explored areas. Then the linear program tries to move robots into unexplored regions by checking the final position of the robots.

$$\begin{aligned} &ieh^x_{ij} = 1,\ where\ r^x_i \geq eh^x_j \\ &ieh^y_{ij} = 1,\ where\ r^y_i \geq eh^y_j \\ &iel^x_{ij} = 1,\ where\ r^x_i \leq el^x_j \\ &iel^y_{ij} = 1,\ where\ r^y_i \leq el^y_j \\ &Robot\ is\ not\ in\ an\ explored\ region,\ i.e. \\ &exploration_{ij} = 1 \\ &where\ ieh^x_{ij} + ieh^y_{ij} + iel^x_{ij} + iel^y_{ij} \geq 1 \end{aligned} \tag{11}$$

So, program gives a final position not located in an explored region. [3]

A sample exploration scenario is shown in Fig. 1(d). Dark (blue) region is explored in the first step, so robots try to locate themselves outside of the explored area.

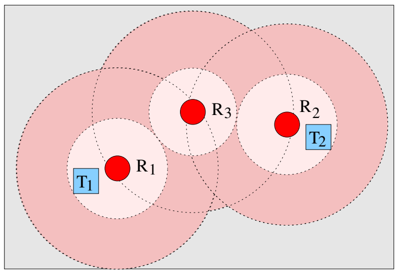

Fig. 3. An example distribution of robots providing optimum target coverage, communication and area coverage. Robot R_1 covers target T_1 and R_2 covers target T_2. R_3 is located to provide communication between them, and its sensing range does not overlap with others. Dark colored circles represent communication range, light colored circles represent sensing range.

G. Optimization Criterion

Optimization criterion is made up of four components, *target coverage*, *communication between robots*, *area covered by the robots* and *the number of robots located in unexplored regions.* Target Coverage: We utilize the number of targets that are covered, i.e.,

$$T = \sum_{j=1}^{n} coverage_j \tag{12}$$

where n=number of targets

Communication: We utilize the number of pairs of robots that can communicate with each other, i.e.,

$$C = \sum_{i=1}^{n}\sum_{j=1}^{n} communication_{ij} \tag{13}$$

where n=number of robots

Area Coverage: We utilize the number of pairs of robots whose sensor ranges do not intersect, i.e.,

$$A = \sum_{i=1}^{n}\sum_{j=1}^{n} area_{ij} \tag{14}$$

where n=number of robots

Exploration: We utilize the number of robots in unexplored regions, i.e.,

$$E = \sum_{i=1}^{n}\sum_{j=1}^{m} exploration_{ij} \tag{15}$$

where n=number of robots, m=number of explored regions.

Optimization Criterion: Our objective function is weighted sum of the above components.

$$maximize\ \alpha T + \beta C + \gamma A + \delta E \tag{16}$$

where α, β, γ, and δ are constants defining priorities.

Figure 3 represents an optimal distribution of robots according to this optimization criterion. Robots arrange themselves so that they cover all targets, provide communication between each other, and cover as much area as possible.

H. Complexity

Our formulation results in mixed-integer linear program, which is NP-Hard in the number of binary variables. However, optimal solutions for reasonable sized problems can be obtained by using commercially available software packages such as CPLEX [21]. Complexity of our program is dominated by the number of binary variables. These are $coverage_j$ for target j, $proximity_{ij}$ for robot i and target j, $communication_{ij}$ and $area_{ij}$ for robots i and j, $exploration_{ij}$, ieh^x_{ij}, ieh^y_{ij}, iel^x_{ij} and iel^y_{ij} for robot i and explored region j, bh^x_{ij}, bh^y_{ij}, bl^x_{ij} and bl^y_{ij} for robot i and obstacle j. For a problem with n targets, m robots, p obstacles and q explored regions, there are $n + nm + 2nn + 5mq + 4mp$ binary variables. So, complexity can be stated as $O(n + nm + n^2 + mq + mp)$.

V. Experiments

We show a sample execution of our program to highlight the properties of the solution. The environment is bounded and has size 12×12. There are three rectangular obstacles, which are located at $\{(0, 4), (5, 6)\}$, $\{(4, 8), (8, 10)\}$ and $\{(8, 2), (10, 6)\}$ (darkest (dark blue) regions in Fig. 4(a)). In the environment there are 8 robots which are located at point $(0, 0)$, and 6 targets whose locations are unknown initially. The targets follow predefined paths and we assume we can predict their locations for the next timestep. Parameters defined for robots and targets are shown at Tables I and II. Timestep is selected to be 4, so robots arrange themselves according to the environment which they estimate to be in 4 steps. In the experiments, we chose constants at optimization criterion as $\alpha > \beta > \gamma > \delta$. In other words, the linear program optimizes (1)*target coverage*, (2)*communication between robots*, (3)*area coverage* and (4)*exploration* from highest to lowest priority, respectively.

Robots start exploring the environment by moving out of the region they explored when they were all at $(0, 0)$. The initial explored region is the rectangle $\{(0, 0), (1, 1)\}$ because the robot with highest sensing range can sense a region of radius 2. Initial locations and environment are shown in Fig. 4(a).

Since there are no targets detected yet, and the communication constraints are satisfied, the robots try to cover as much area as possible while obeying the movement constraints. The new environment is shown in Fig. 5(a) where blue (darker) areas indicate explored regions. Exploration reveals targets t_1 and t_2, and predicts their positions to be $(0, 4)$ and $(2, 2)$, respectively. Optimal allocation is shown in Fig. 5(b). Robots r_6 and r_8 cover targets, and other robots continue exploration while staying within the communication range. Next, target t_3 is found, which requires two robots to be covered. Robots r_2, r_3 and r_7 continue exploration and r_6 works as the communication bridge while remaining robots are assigned to the targets. Distribution of robots is shown in Fig. 5(c). Two other targets, t_4 and t_5 are discovered

[3]Please see the appendix for the proof that given sufficient number of robots for communication and target tracking, our algorithm will result in the exploration of the all environment.

TABLE I
DEFINED PARAMETERS FOR ROBOTS

	r_1	r_2	r_3	r_4	r_5	r_6	r_7	r_8
Sensing Range	1	2	1	1	2	1	2	1
Robot Speed	1	2	2	1	1	1	2	2
Comm. Range	4	4	4	4	4	4	4	4

at the next step. Moreover, targets t_1 and t_2 move faster than their controller robots, r_1 and r_4, which cannot catch them. However, global optimization finds a solution to this problem by assigning the tracking task to other robots that can reach the targets (Fig. 5(d)). Target t_6 is discovered at the next step. At this time, it is not possible to cover all the targets while keeping the communication between all robots. Since target coverage is given more importance, robots are distributed into two independent groups. Robots r_3 and r_5 form one team, while others form the other team. Each team has communication in itself, but cannot reach to the other team. An optimal solution is found and applied for each team. Fig. 5(e) represents result of two optimal solutions. Targets t_1 and t_5 leave the environment at the next step. Team of robots r_3 and r_5 has one target to follow, so while one robot follows target, the other robot, in this case r_3, which is the faster robot, continues exploration. The other team covers all targets, and provides communication in itself. Fig. 5(f) shows the final state of the environment which is totally explored.

We have also experimented with the effects of eliminating some of the components from the objective function under the same experiment scenario. Figs. 4(b) and (c) show the final configuration at the end of the simulation where the area coverage and exploration components are removed from the objective function, respectively. In both of these experiments, not all of the targets were tracked or all environment was explored because of the decrease in effectiveness of the method when some functions are disabled. In Fig. 4(b) robots stand together whenever they do not need to cover a target or provide communication between target covering robots. Exploration does not compensate this problem because several robots can move to the same region, which all of them consider as unexplored. This has a drastic effect on the explored area and covered targets, so performance is considerably lower than the original formulation. In this scenario, targets t_4 and t_6 remain undiscovered and upper part of the environment remains unexplored. In Fig. 4(c), performance of the system is better, because the environment is small and maximum area coverage helps exploration when robots are following targets. However, robots have no motivation to move unless targets drag them, which can leave some parts of the environment totally unexplored. In this example, target t_4 remains undiscovered, and some region still remains unexplored although it is smaller this time.

Our experiment shows that we can succesfully assign tasks to the robots. We can successfuly track individual targets, keep communication distance as long as possible, provide maximum area coverage and explore the environment.

TABLE II
DEFINED PARAMETERS FOR TARGETS

	t_1	t_2	t_3	t_4	t_5	t_6
Cov. Requirement	1	1	2	1	1	1

VI. EXTENSIONS

Linear programming provides a powerful modelling tool. Variations in the problem can be stated easily by modifying objectives and constraints. We will mention a few of the modifications that can be applied to our program. In multi-robot systems, energy consumption is an important problem. One way for achieving optimal energy usage is by adding a new objective function to the optimization criterion, which minimizes the total distance covered by all robots.

$$D = -\sum_{i=1}^{n} movement_i \tag{17}$$

where n=number of robots.

Another important energy concern is the usage of wireless communication, which can have drastic effect on small robots. Less power can be used for shorter range communication, so minimizing distance between robots can reduce communication cost.

$$CE = -\sum_{i=1}^{n}\sum_{j=1}^{n} distance_{ij}^{RR} \tag{18}$$

where n=number of robots.

Initially, we assumed it is enough for targets to be in sensing range of some robots to be considered as covered, but application may require robots to be as close as possible to targets.

$$TD = -\sum_{i=1}^{n}\sum_{j=1}^{m} distance_{ij}^{RT} \tag{19}$$

where n=number of robots, m=number of targets.

In the original formulation, we assumed a robot can cover more than one target. The constraint that allows each robot to cover a single target can also be enforced.

$$\sum_{j=1}^{m} proximity_{ij} \leq 1 \; for\ each\ robot\ i \tag{20}$$

where m=number of targets.

Robots can have constraints imposed by non-holonomic constraints. For example, if a robot is moving in a direction, it may require some time for it to go into reverse direction. This can be enforced by restricting the robot going into places beyond its limit. Assume that the robot is moving in $+y$ direction, then a contraint can be:

$$r_i^y - ir_j^y \geq |r_i^x - ir_j^x| * maneuver\ capability \tag{21}$$

This constraint can be changed according to the current direction and maneuver capability of the robot.

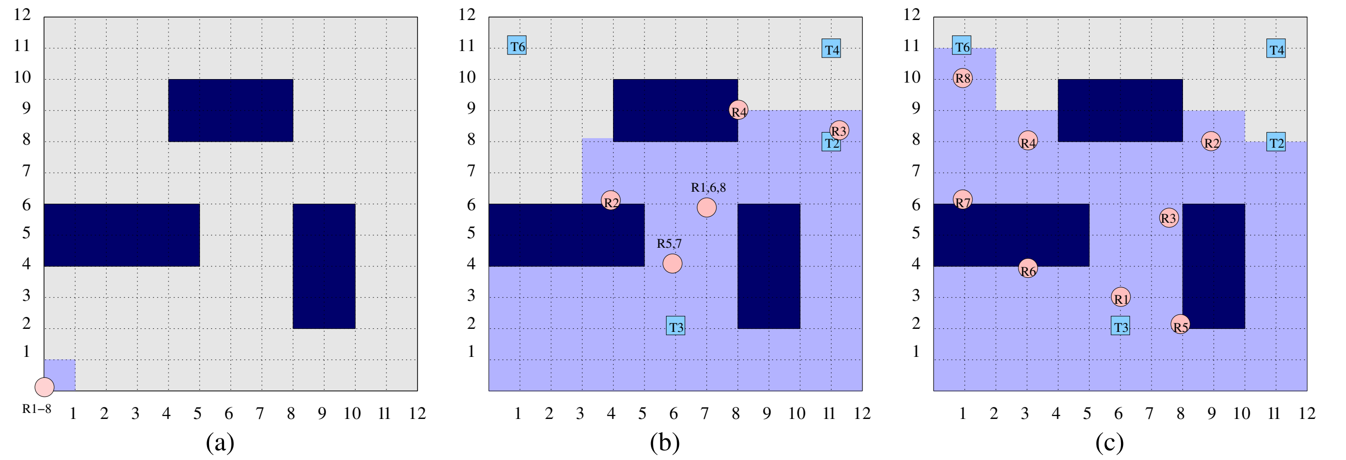
Fig. 4. (a) Initial configuration of the environment and robots. Robots are represented as circles, and targets are represented as squares. (b) Final configuration when area coverage is not optimized. (c) Final configuration when exploration is not optimized.

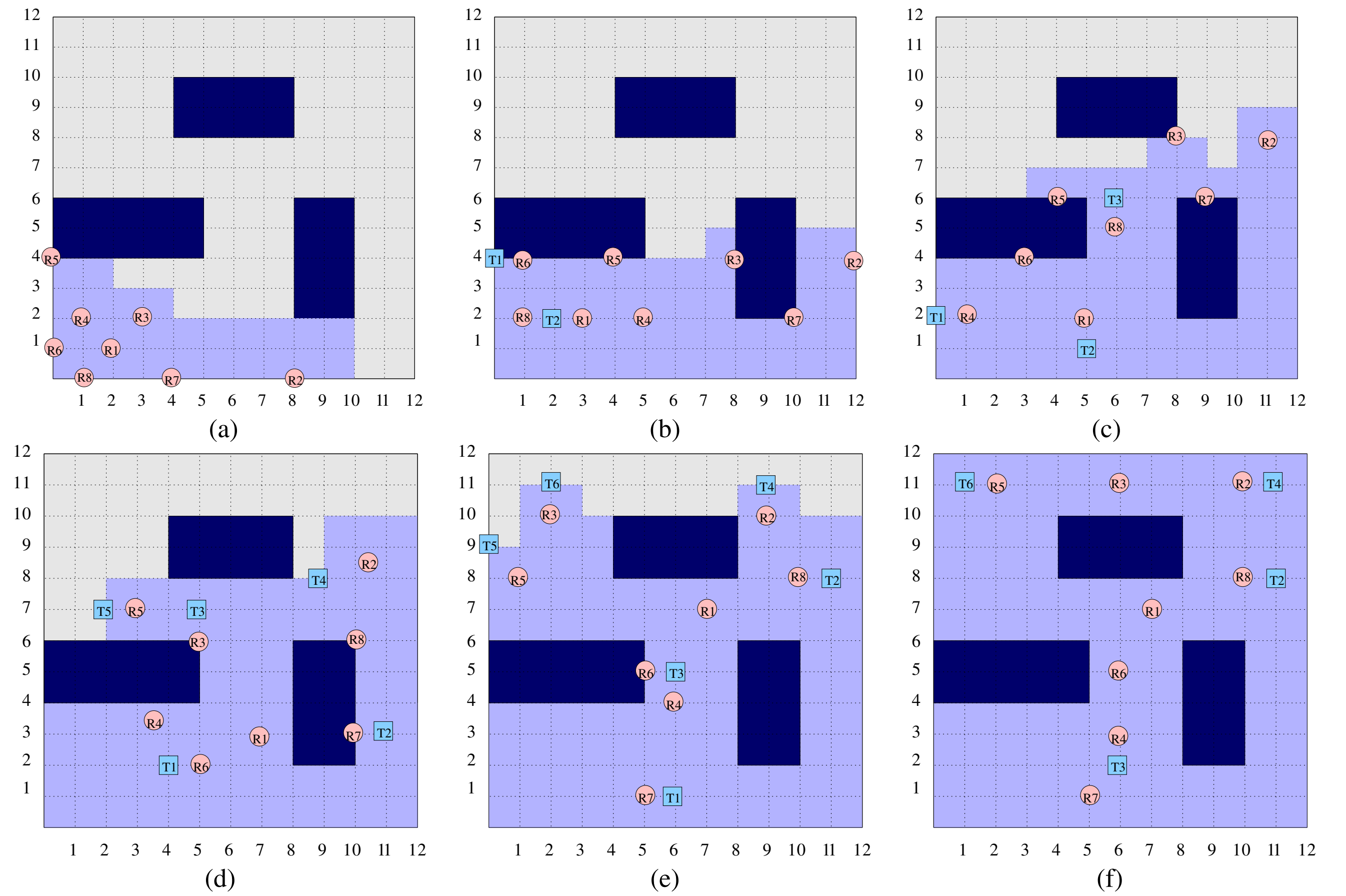
Fig. 5. Sample execution of the linear programming method. Robots are represented as circles, and targets are represented as squares. Dark blue (darkest) regions are obstacles, blue (darker) regions are explored regions, and gray (light gray) regions are unexplored regions.

VII. Conclusions and Future Work

We have presented a mixed-integer linear programming method to solve the task allocation problem of multiple heterogeneous robots for detecting and controlling multiple regions of interest in an unknown environment under defined constraints. The method presented here finds a global optimal solution for the given state of the robots, targets and environment. We presented our assumptions, constraints and linear program formulations. This linear program gives the optimal locations of the robots to maximize the objective function while obeying constraints. Some of the alternatives to objective function and constraints are also represented to show the flexibility of the proposed method. We verified validity and efficiency of our approach using simulation results.

Our future work includes developing a fully distributed version of this method. One obvious problem with mixed-integer linear programming is scalability because of the increase in complexity with the increased number of variables. However, with the current processing power that mobile robots have, wireless communication can be more limiting especially when multi-hop communication is required. We are currently working on developing a distributed version of our solution, where robots find local solutions, and exchange data to optimize solution as needed.

Appendix

Absolute value function is not a linear function. However, it is possible to model it using linear constraints. One common way for doing this is defining two extra variables.

$$x = x^{+} - x^{-} \; and \; |x| = x^{+} + x^{-}$$
$$where \; x^{+} \geq 0 \; and \; x^{-} \geq 0$$
$$minimize \; |x|$$

This formulation gives $|x|$ as the absolute value of x. In our program, we did not need to use minimization step because exact values of variables are not important as long as those values are below some constant.

Theorem 1: Robots cover all targets as long as targets move slower than robots if target coverage has the highest priority.

Proof: A target t_j is detected when it is in sensing range of a robot r_i. In other words, $distance(r_i^t, t_j^t) \leq sensingRange_i$, where r_i^t is the position of robot r_i, and t_j^t is the position of target t_j at time step t. Our assumption requires that robot speed is greater than or equal to the target speed, so at time step $t+1$, $distance(r_i^{t+1}, r_i^t) \geq distance(t_j^{t+1}, t_j^t)$. The highest priority is given to target coverage in linear program formulation, so robot r_i will be assigned to cover t_j unless other robots cover it. Target t_j can move furthest if it moves on the line connecting r_i and t_j, in the opposite direction of robot. In that case,

$$distance(r_i^{t+1}, t_j^{t+1}) =$$
$$distance(t_j^{t+1}, t_j^t) + distance(r_i^t, t_j^t) - distance(r_i^{t+1}, r_i^t) \leq$$
$$distance(r_i^{t+1}, r_i^t) + distance(r_i^t, t_j^t) - distance(r_i^{t+1}, r_i^t) =$$
$$distance(r_i^t, t_j^t) \leq sensingRange_i.$$

■

Theorem 2: Robots explore a bounded environment in finite time if there are more mobile robots than needed to cover targets and provide communication between covering robots.

Proof: Assume that there are k mobile robots which are not assigned for covering targets or providing communication in a bounded environment of size $w * h$, where w is the width and h is the height of the environment. Each of these robots has sensing range and speed which are greater than 0. Assume each of them has uniform sensing range r and speed s. These robots can be located inside the explored region, or on the border between explored and unexplored regions. If they are located inside explored region, they can move to unexplored region in at most $\frac{(w+h)}{s}$ timesteps. The linear program locates them in unexplored region at each step if robots can reach unexplored regions, exploring a region of size at least r. Since the environment is bounded of size $w{*}h$, environment will be totally explored in $\frac{(w+h)}{s} + \frac{(w*h)}{r}$. ■

References

[1] R. Davis and R. G. Smith, "Negotiation as a metaphor for distributed problem solving," *Artificial Intelligence*, vol. 20, pp. 63–109, 1983.

[2] B. P. Gerkey and M. J. Matarić, "Sold!: Auction methods for multi-robot coordination," *IEEE Transactions on Robotics and Automation*, vol. 18, no. 5, pp. 758–786, October 2002.

[3] S. Botelho and R. Alami, "M+: a scheme for multi-robot cooperation through negotiated task allocation and achievement," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Detroit, Michigan, May 1999, pp. 1234–1239.

[4] R. Zlot and A. Stentz, "Complex task allocation for multiple robots," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Barcelona, Spain, April 2005, pp. 1515–1522.

[5] G. Thomas, A. M. Howard, A. B. Williams, and A. Moore-Alston, "Multi-robot task allocation in lunar mission construction scenarios," in *IEEE International Conference on Systems, Man and Cybernetics*, vol. 1, Hawaii, October 2005, pp. 518–523.

[6] T. Lemaire, R. Alami, and S. Lacroix, "A distributed tasks allocation scheme in multi-uav context," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, New Orleans, LA, April 2004, pp. 3822–3827.

[7] L. E. Parker, "Alliance: An architecture for fault tolerant multirobot cooperation," *IEEE Transactions on Robotics and Automation*, vol. 14, no. 2, pp. 220–240, April 1998.

[8] B. B. Werger and M. J. Matarić, "Broadcast of local eligibility for multi-target observation," in *5th International Symposium on Distributed Autonomous Robotic Systems (DARS)*, Knoxville, TN, October 4-6 2000, pp. 347–356.

[9] B. P. Gerkey and M. J. Matarić, "Multi-robot task allocation: Analyzing the complexity and optimality of key architectures," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Taipei, Taiwan, September 17-22 2003, pp. 3862–3867.

[10] ——, "A formal analysis and taxonomy of task allocation in multi-robot systems," *Intl. Journal of Robotics Research*, vol. 23, no. 9, pp. 939–954, September 2004.

[11] K. Nygard, P. Chandler, and M. Pachter, "Dynamic network flow optimization models for air vehicle resource allocation," in *The American Control Conference*, Arlington, Texas, June 25-27 2001, pp. 1853–1858.

[12] J. Bellingham, M. Tillerson, A. Richards, and J. How, "Multi-task allocation and path planning for cooperating uavs," pp. 1–19, November 2001.

[13] C. Schumacher, P. Chandler, M. Pachter, and L. Pachter, "Uav task assignment with timing constraints," in *AIAA Guidance, Navigation, and Conference and Exhibit*, Arlington, Texas, 2003.

[14] Y. Jin, A. Minai, and M. Polycarpou, "Cooperative real-time search and task allocation in uav teams," in *42nd IEEE Conference on Decision and Control*, Maui, Hawaii USA, December 2003, pp. 7–12.

[15] C. Schumacher, P. Chandler, S. Rasmussen, and D. Walker, "Task allocation for wide area search munitions with variable path length," in *The American Control Conference*, Denver, Colorado, June 2003, pp. 3472–3477.

[16] M. Alighanbari, Y. Kuwata, and J. How, "Coordination and control of multiple uavs with timing constraints and loitering," in *The American Control Conference*, vol. 6, Denver, Colorado, June 4-6 2003, pp. 5311–5316.

[17] D. Turra, L. Pollini, and M. Innocenti, "Fast unmanned vehicles task allocation with moving targets," in *43rd IEEE Conference on Decision and Control*, Atlantis, Paradise Island, Bahamas, December 14-17 2004, pp. 4280–4285.

[18] P. B. Sujit, A. Sinha, and D. Ghose, "Multi-uav task allocation using team theory," in *44th IEEE International Conference on Decision and Control, and the European Control Conference*, Seville, Spain, December 12-15 2005, pp. 1497–1502.

[19] M. A. Darrah, W. Niland, and B.M.Stolarik, "Multiple uav dynamic task allocation using mixed integer linear programming in a sead mission," in *Infotech@Aerospace*, Arlington, Virginia, September 26-29 2005.

[20] A. Elganar and K. Gupta, "Motion prediction of moving objects based on autoregressive model," *IEEE Transactions on Systems, Man and Cybernetics-Part A:Systems and Humans*, vol. 28, no. 6, pp. 803–810, November 1998.

[21] *ILOG CPLEX 9.0 User's Manual*, ILOG, Inc., http://www.ilog.com/products/cplex/, October 2003.