Multi-robot Terrain Coverage and Task Allocation For Autonomous Detection of Landmines

Prithviraj Dasgupta¹, Angelica Muñoz-Meléndez¹,² and K. R. Guruprasad¹,³

¹ Computer Science Department, University of Nebraska at Omaha, NE, USA
² Computer Science Dept., National Institute of Astrophysics, Optics and Electronics, Mexico
³ Mechanical Engg. Department, National Institute of Technology, Karnataka, India

E-mail: {pdasgupta, amunozmelendez, gkannarpadi}@unomaha.edu

ABSTRACT

Multi-robot systems comprising of heterogeneous autonomous vehicles on land, air, water are being increasingly used to assist or replace humans in different hazardous missions. Two crucial aspects in such multi-robot systems are to: a) explore an initially unknown region of interest to discover tasks, and, b) allocate and share the discovered tasks between the robots in a coordinated manner using a multi-robot task allocation (MRTA) algorithm. In this paper, we describe results from our research on multi-robot terrain coverage and MRTA algorithms within an autonomous landmine detection scenario, done as part of the COMRADES project. Each robot is equipped with a different type of landmine detection sensor and different sensors, even of the same type, can have different degrees of accuracy. The landmine detection-related operations performed by each robot are abstracted as tasks and multiple robots are required to complete a single task. First, we describe a distributed and robust terrain coverage algorithm that employs Voronoi partitions to divide the area of interest among the robots and then uses a single-robot coverage algorithm to explore each partition for potential landmines. Then, we describe MRTA algorithms that use the location information of discovered potential landmines and employ either a greedy strategy, or, an opportunistic strategy to allocate tasks among the robots while attempting to minimize the time (energy) expended by the robots to perform the tasks. We report experimental results of our algorithms using accurately-simulated Corobot robots within the Webots simulator performing a multi-robot, landmine detection operation.

1. INTRODUCTION

Humanitarian demining is a crucial effort for the safety and sustenance of human lives in post-conflict regions. Unfortunately, recent surveys on landmine monitoring report that humanitarian demining efforts are considerably lagging behind anti-personnel landmine planting activities due to several technological and economic reasons.¹ This results is enormous loss to human lives. One the major technological challenges in humanitarian demining is to detect landmines rapidly and with reasonable accuracy, while reducing the number of false positives. We envisage that automating landmine detection operations using multiple, off-the-shelf autonomous robots will provide a reasonably accurate yet economical solution to the problem of detecting landmines. Towards this objective, we are developing the COMRADE (COoperative Multi-Robot Automated DEtection) system for humanitarian demining. The central objective of the COMRADE system is to develop novel coordination techniques between multiple low-cost, autonomous, mobile robots, which enable them to collaboratively detect landmines with high accuracy in post-conflict regions. COMRADES includes techniques that will allow each robot to explore an initially unknown region while searching for landmines, recognize landmine-like objects on its sensors and coordinate its actions with other robots so that multiple robots with different types sensors can converge on the object to analyze and confirm it as a landmine.

¹In 2010 alone, explosions of landmines and similar devices resulted in 4,191 casualties, with civilians accounting for 70% of the casualties.
2. DESCRIPTION OF THE COMRADE SYSTEM

The main hypothesis of the COMRADE system is that the detection accuracy of sub-terrain landmines can be improved if the analysis of the data collected from the same object or region is performed by multiple types of sensors. A major thrust in the recent efforts for automated demining using robots has focused on developing and refining the design and construction of robots suitable for detecting landmines. These robots are quite expensive to manufacture and deploy. Moreover, the landmine detections reported using individual robots is prone to a large number of false positives due to detection of non-landmine sub-terrain objects with landmine-like compositional characteristics. To address these issues, we posit that the process of landmine detection can be performed accurately yet economically if low-cost robots equipped with different types of sensors for landmine detection can be autonomously coordinated to detect, collect and analyse the data from landmine-like objects. The principal features of the COMRADE system which achieves this objective is shown in Figure 1.

![Figure 1. General schematic of COMRADE system for landmine detection using multiple, autonomous, mobile robots](image)

The robots used in the COMRADE system are off-the-shelf, relatively inexpensive, autonomous robots. These robots have to be outfitted with appropriate sensors that are capable of detecting landmines. The costs, accuracy and capabilities of different sensors in the COMRADE system are given in Table 1. Operational costs of each type of sensor is also proportional to its procurement cost.

Because of the widely varying costs of different sensor types in COMRADES, it is important to consider the habitation and the risks to human lives in the area of interest before deploying the robots. Consider, for example, three types of regions - urban and semi-urban areas, agricultural areas, and uncultivated, infrequented areas. Urban and semi-urban areas, although densely inhabited, have a low chance of having buried landmines because the terrain is likely to have been considerably agitated while constructing human habitation. Therefore, in such areas the risk to lives from landmine explosion is considerably low. In contrast, agricultural land has
frequent human traffic and is less inhabited. The terrain of agricultural regions is also likely to have been less agitated than urban areas. Therefore, in agricultural regions risks to humans lives from landmine explosion is high. Finally, regions that are uncultivated and not frequented by humans might have a high chance of having a landmine, but the risk to human lives is moderate as these regions are not frequented often. Based on the different risk factors and possibility of existence of landmines in different types of regions, in the COMRADE system, the area of interest (AOI) is classified (by humans) before deploying robots.

<table>
<thead>
<tr>
<th>Sensor Type</th>
<th>Cost (USD)</th>
<th>Accuracy / Materials Detected</th>
</tr>
</thead>
<tbody>
<tr>
<td>Metal Detector (MD)</td>
<td>$100 – $200</td>
<td>Low / only metal</td>
</tr>
<tr>
<td>Infra-red based Multi-detector (IR)</td>
<td>$2,000 – $5,000</td>
<td>Moderate / metal and plastic</td>
</tr>
<tr>
<td>Ground penetrating radar (GPR)</td>
<td>$20,000 – $30,000</td>
<td>High / metal and plastic</td>
</tr>
</tbody>
</table>

Table 1. Costs of different types of landmine detection sensors in the COMRADE system.

<table>
<thead>
<tr>
<th>Type of AOI</th>
<th>Risk category</th>
<th>Number and Type of robots</th>
</tr>
</thead>
<tbody>
<tr>
<td>High traffic, low chance of landmine</td>
<td>Low risk</td>
<td>Individually moving robots, mainly with MD</td>
</tr>
<tr>
<td>Moderate traffic, high chance of landmine</td>
<td>High risk</td>
<td>Tightly coupled robot teams with most MD, some IR and few GPR sensors</td>
</tr>
<tr>
<td>Low traffic, high chance of landmine</td>
<td>Moderate risk</td>
<td>Individually moving robots with most MD, some IR and few GPR sensors</td>
</tr>
</tbody>
</table>

Table 2. Classification of different types of AOI in COMRADES depending on risk to human lives.

The number, type and coordination between robots deployed into the AOI is based on the AOI’s classification, as described in Table 2. Based on the costs of different types of sensors given in Table 1, it makes sense to use high cost sensors only in regions where they are essential, that is, in regions where there is higher risks to human lives. In low risk regions, many robots with low cost MD sensors can be used to detect landmines. At the same time, in high risk regions, it is essential to confirm an object detected as a potential landmine very rapidly so that human lives are not endangered during the time the confirmation process is ongoing. Therefore, in such a setting, it makes sense to keep robots with different types of sensors in close proximity of each other as a tightly coupled team, so that robots with different types of sensors can be made available immediately when any one sensor type records a signature of a potential landmine. In contrast, in low risk regions robots can move individually, while in regions with intermediate risk, robots can move in such a way that robots with different sensor types can be assimilated within a certain guaranteed time to confirm a potential object as a landmine.

To investigate our hypothesis, our research in developing the COMRADE system focusses on two major technical objectives - a) distributed terrain or area coverage by multiple robots to detect location of potential landmines, and b) multi-robot task allocation between robots to collectively confirm the location and characteristics of objects identified as potential landmines. In the rest of the paper, we describe the techniques used in the COMRADE system to achieve these technical objectives.

3. DISTRIBUTED TERRAIN COVERAGE IN COMRADES

Distributed area or terrain coverage of an initially unknown environment using a multiple robot system (MRS) is a central aspect of COMRADES. Distributed coverage offers several advantages over using a single robot to perform coverage, such as reduced time to complete coverage and improved robustness against single or multiple robot failures. However, a challenging problem in distributed coverage using a MRS is to achieve decentralized coordination between different robots so that they do not impede each others’ movement or perform repeated
coverage of regions within the environment. Most previous approaches to multi-robot distributed coverage assume that the environment is decomposed into a cellular or grid-like structure before deploying the robots, and focus on coverage algorithms that allow the robots to visit these cells while trying to achieve desirable outcomes such as completeness and non-redundancy. However, the desired operation of most of these coverage algorithms relies crucially on the underlying (exact or approximate) cellular decomposition. This decomposition is usually either assumed to be done off-line, or, the space is decomposed into conveniently sized cells that are aligned with the footprint of a robot. However, both these approaches have certain disadvantages - the cellular decomposition technique of the environment might not usually be truly distributed or, having cell size match robot’s footprint might lead to increased computation and planning by the robots before each movement. Therefore, it makes sense to investigate distributed coverage techniques that can handle both decomposition and coverage of the environment in a distributed manner.

In COMRADES, we use Voronoi partition to spatially partition the multi-robot coverage task. We have used an algorithm called Voronoi Partition-based Coverage (VPC) which is described below.

Let $Q \in \mathbb{R}^2$ denote the environment or search space. The boundary $\partial Q$ of the search space $Q$ is known to the robots. Let $D \subset Q$ be the part of $Q$ containing obstacles. There is no a priori information about $D$. The region $Q \setminus D$ is required to be completely covered by $N$ mobile autonomous robots which are equipped with sensors/tools. The area swept by a robot’s sensor/tool is called the area covered by a robot. The complete coverage task for $N$ robots is to move in $Q \setminus D$ such that the union of the swept areas of all robots’ sensors/tools is equal to $Q \setminus D$. In a space which is discretized into cells based on the sensor/tool footprint, complete coverage task is achieved if each cell is visited by at least one robot. The coverage is non-overlapping if no region (cell, in a discretized space) is covered more than once.

Voronoi partition is a widely used scheme for partitioning a space. Let $I_N = \{1, 2, \ldots, N\}; Q \subset \mathbb{R}^2$, be a convex polygon; and $\mathcal{P} = \{p_1, p_2, \ldots, p_N\}, p_i \in Q$, be a set of points in $Q$ called a node set. The Voronoi partition, generated by $\mathcal{P}$ is the collection $\{V_i(\mathcal{P})\}_{i \in I_N}$ with,

$$V_i(\mathcal{P}) = \{q \in Q| \parallel q - p_i \parallel \leq \parallel q - p_j \parallel, \forall j \in I_N\}$$

(1)

The Voronoi cell $V_i$ is the collection of those points which are closest to $p_i$.

Let $p_i(t) \in Q \setminus D$, be position of $i$-th robot at time $t$. Each robot constructs its Voronoi cell using $\mathcal{P}(0)$ as the node set. Now the coverage task for $i$-th robot is to completely cover $V_i \setminus D$, portion of its Voronoi cell free of obstacles. Thus a multi-robot coverage task is now partitioned into $N$ single robot coverage tasks.

### 3.1 Voronoi Partition-based Coverage Algorithm

The algorithm executed by a robot for performing Voronoi Partition-based Coverage is shown in Algorithm 1. Each robot first exchanges its position with all other robots so that each robot can compute its Voronoi cell $V_i$. Robot $i$ then covers the region within its assigned Voronoi cell using a coverage algorithm such as STC. While covering it exchanges alive messages with its Voronoi neighbors. After completing coverage, robot $i$ checks if it has been receiving alive messages from all its Voronoi neighbor. If such a neighbor exists and it has not yet completed its coverage, it is marked as a failed robot and the Voronoi cells are recomputed while discarding the failed robot. Robot $i$ then proceeds to cover the region of its new Voronoi cell (excluding already covered portion) to ensure that the region left incompletely covered by the failed robot is covered.

It can be shown that the VPC algorithm ensures complete non-overlapping coverage provided the single-robot coverage algorithm achieves complete non-overlapping coverage. Further, it can also be shown that the VPC algorithm is robust to failure of some of the robots and also to localization errors.

### 3.2 Results of Simulation Experiments

We have verified the VPC algorithm on the Webots simulator using a model of the Coroware Corobot robot, as illustrated in Figure 2. The Corobot robot is equipped with a four-wheel drive base, four infra-red sensors - two located in the front and two on the sides of the robot, for avoiding collisions, and, an indoor GPS called a Stargazer kit for localization. The robots are able to communicate wirelessly within the environment that measures $20 \times 20$ m.
VPC \((Q, I, P_i(0))\)

**Input:** \(Q, I, P_I(0)\) \(\triangleright Q:\) search space, \(I:\) set of robots, \(P_I(0) = \{p_1(0), p_2(0)\ldots\}\): initial location of each robot \(i \in I\)

**Output:** void

\[
\text{aliveTimer} \leftarrow 0, \text{coverageCompleted} \leftarrow \text{false};
\]

Broadcast \(p_i(0)\) to every \(j \in I \setminus \{i\}\);

\[V_i \leftarrow \text{computeVoronoiCell}(P_I(0), Q);\]

// \(V_i\) is the Voronoi cell for robot \(i\)

// \(N_i\) is the set of Voronoi neighbors of robot \(i\)

while \(\text{true}\) do

\[
\text{if } \text{completedCoverage} = \text{false} \text{ then}
\]

Perform motion (action) prescribed by coverage algo (e.g., STC) to cover \(V_i \setminus D\)

\[
\text{if } \text{coverage algo signals coverage completion} \text{ then}
\]

\[
\text{coverageCompleted} \leftarrow \text{true};
\]

\[
\text{sendMessage(“Completed Coverage”) to every robot } j \in N_i
\]

\[
\text{end}
\]

\[
\text{aliveTimerUpdate();}
\]

\[
\text{end}
\]

\[
\text{else}
\]

\[
\text{if } \text{coverageCompleted} = \text{true}, \text{forall } j \in I \setminus i \text{ then}
\]

\[
\text{STOP;}
\]

\[
\text{end}
\]

\[
\text{else}
\]

checkForFailedNeighbors();

\[
\text{if } \exists \text{ failedNeighbors that did not complete coverage} \text{ then}
\]

\[
\text{coverageCompleted} \leftarrow \text{false};
\]

\[
\text{end}
\]

\[
\text{else}
\]

\[
\text{aliveTimerUpdate();}
\]

\[
\text{end}
\]

\[
\text{end}
\]

**Algorithm 1:** Voronoi Partition-based Coverage Algorithm for robot \(i\)

Figure 2. Robot platform used for testing coordination mechanisms for task sharing. (a) Simulated robot in the environment Webots and (b) physical Corobot manufactured by Coroware.

Individual robots cover their Voronoi cells using the on-line single robot STC algorithm. Figures 3(a) and 3(b) show the coverage of a square region and a X-shaped region having obstacles (unknown to robots). Regions in the boundary of Voronoi cells are not covered, as STC algorithm (which is used here work only to demonstrate VPC strategy) considers most cells in boundary as occupied with obstacles. Figure 3(c) shows coverage of the Voronoi cell of a failed robot is completed by its Voronoi-neighbors, thus completing the coverage.
Figure 3. 4 robots cover a) a 20m × 20m square environment, and b) a X-shaped region within this environment using VPC algorithm. (c) One robot fails in the same setting as in scenario (a), and an operational robot takes over coverage of the failed robot’s Voronoi cell.

Figure 4. Illustration of Voronoi cell computation using DVC. A candidate Voronoi cell $V_3$ is formed at the end of expansion phase and the final Voronoi cell $V$ is computed in the contraction phase. The candidate Voronoi cell at each step is shown by the region enclosed by bold lines.

3.3 Distributed computation of Voronoi cell

In the VPC algorithm described in Section 3.1, each robot needs to compute only its corresponding Voronoi cell. However, each robot computes the entire Voronoi partition, extracts its Voronoi cell and discards the information about the remaining cells. The discarded information corresponds to considerable amounts of useless computation done by each robot, and incurs unnecessary expenditure of energy and time. We are currently developing a distributed algorithm for computing Voronoi cell. In this technique, each robot represents other robots position in a polar coordinate system relative to its own position. The Voronoi cell is computed in two distinct phases, namely, expansion phase and contraction phase. In the expansion phase, a virtual sensor range of the robot is successively increased in steps until the Voronoi cells based on positions of robots within the current virtual range is a polygon. Figure 4 illustrates the process with the help of an example. At the end of the expansion phase, a candidate Voronoi cell is formed. This candidate Voronoi cell is further refined in the contraction phase, where nodes (position of robots) are checked for possible contraction of the candidate Voronoi cell.

In most practical situations for robotic coverage, robots may not have information about the positions of all other robots in the environment, e.g., when subsets of robots are outside each others’ communication range. In such a situation, it is not possible to compute the exact Voronoi cells for each robot using the distributed algorithm described above. To address this problem, we are developing a distributed algorithm for calculating each robot’s Voronoi cell constrained by the robot’s sensor range. An illustration of the computation of the range-constrained Voronoi cell using the distributed Vornoi cell computation algorithm is shown in Figure 5.

4. DISTRIBUTED MULTI-ROBOT COORDINATION FOR TASK SHARING

In parallel with the exploration phase, the robots in COMRADES need to coordinate their movement with each other so that robots with different types of sensors can converge at the location of an object that could possibly
be a landmine, and share the results of data analyses performed using their individual sensors to predict with reasonable accuracy, if the object is a landmine. The coordination between the robots is a crucial aspect to achieve this objective. We have developed suitable Multi-Robot Task Allocation (MRTA) techniques that allow robots to share their sensor data related to the detection of potential landmine-like objects with each other and to make decisions about their movement and actions so that robots equipped with different types of mine detection sensors can converge at objects that potentially appear as landmines and confirm them in an efficient manner.

We consider a category of MRTA problems called ST-MR-TA (single task robot, multi-robot tasks, time extended assignment), where ST stands for single-task robots, i.e., each robot is able to execute as most one task at a time, MR means multi-robot tasks, tasks that require multiple robots to be completed, and TA means time-extended assignment, problems where the information to allocate tasks to robots arrives over time.

The problem of multi-robot task allocation (MRTA) has been investigated using different techniques, and, recently with market-based approaches. One of the earliest systems using for MRTA was the M+ system. The traderbots approach by uses multi-round, single-item auctions for dynamic task allocation across multiple robots, while in the traderbots approach is augmented using the Skill, Tactics, Play (STP) approach for coordinated teamwork. The MRTA problem has also been approached as an exploration problem of matching a set of robots to a set of targets using an algorithm called PRIM-ALLOCATION. Zlot and his team have also used auction-based algorithms for multi-robot task allocation. The MRTA problem has also been combined with techniques from multi-agent coordination and optimization techniques such as negotiation, coalition formation, reinforcement learning, vector regression learning, Hungarian algorithm, vacancy chains, and dynamic vehicle routing to improve the performance of the robots and deal with uncertainty. The general problem of finding an optimal allocation of tasks for a set of machines that satisfies a given criterion of optimality, such as minimizing the completion time is a complex combinatorial problem. If the problem concerns machines like mobile robots that usually have to deal with uncertain and dynamic environments, the problem might become unmanageable even for a set of dozens of tasks. Below we describe the models investigated so far to enable the robots in COMRADES to jointly search for landmine-like objects in a decentralized manner.

### 4.1 Models for Multi-Robot Task Allocation

Let $Q \subset \mathbb{R}^2$ represent a bounded 2-D environment and $R = \{r_i : 1 \leq i \leq m\}$ represent a set of $m$ mobile robots that are deployed within $E$. $p_{r_i}(t) \in Q$ denotes the position of robot $r_i$ at time $t$. There are $n$ stationary targets corresponding to potential landmines, which are distributed within the environment. Each target requires a subset of robots in $R$ to operate upon it. The set of operations performed by different robots on a target is referred to as a task. Let $T = \{\tau_i : 1 \leq i \leq n\}$ represent a set of tasks. Each task $\tau_i$ is associated with four attributes: its position in the environment $p_{\tau_i} \in E$, a demand value $nd_{\tau_i} \in \mathbb{Z}$ that denotes the number of
robots that need to operate on the task to complete it, a progress value $ad_{r_i} \in \mathbb{Z}$ that denotes the number of robots that have already serviced the task, and, a Boolean availability value $avail_{r_i}$ denoting whether the task is currently being serviced by a robot and is consequently unavailable. Let $T_{open} = \{ \tau_i \in T : ad_{r_i} < nd_{r_i} \}$, $T_{closed} = T \setminus T_{open}$, and, $T_{avail} = \{ \tau_i \in T : avail_{r_i} = true \}$ represent the sets of open, closed and available tasks respectively. $d_{i,j} = | || p_{r_i} - p_{r_j} ||$ is the Euclidean distance between tasks $\tau_i$ and $\tau_j$ and $\hat{d}_{i,j} = | || p_{r_i}(t) - p_{r_j} ||$ is the Euclidean distance between robot $r_i$ at time $t$ and task $\tau_j$. When a robot reaches the location of a task, one unit of the task’s demand is processed and its progress increases by 1. A task is completed when its progress matches its demand, i.e., when $ad_{r_i} = nd_{r_i}$.

4.1.1 Greedy Distance based Model

The greedy model is a well-known mechanism that allocates tasks considering only an immediate reward. In our greedy approach incoming requests of service are released by a call center to the robot team, and tasks are allocated to the nearest available robot, on a first-come first-served basis. The greedy algorithm allocates tasks to robots based on the distance between robots and demands as the first criterion, and as a second criterion in the numerical identifier of robots to solve eventual conflicts that cannot be solved by the former.

The advantage of a greedy model is its simplicity, a robot using a greedy model for selecting asks does not invest resources to gaze on the horizon. Sometimes however more sophisticated solutions are required to find a good or acceptable solution.

4.1.2 Stochastic Queueing based Model

Our second MRTA model is based on spatial queueing theory. In this model, the demands at different targets are generated stochastically depending on the availability of target. A solution to the problem consists of each robot calculating an ordered sequence of demands based on the costs to process demands while minimizing certain metrics such as the distance traveled by the robots to process demands, or, the waiting/idle time for targets. When the spatial distribution of demands (targets) in the environment is known, a queueing approach can be enriched applying a spatial framework. These systems generally evolve over time as Markovian processes and the robots select tasks according to a Markovian mechanism.

We represent the probability of a robot to select task $\tau_j$ after it has serviced task $\tau_i$ as an inter-task transition matrix $M_{\tau}$ given by:

$$M_{\tau} = \begin{pmatrix} \pi_{11} & \pi_{12} & \cdots & \pi_{1n} \\ \pi_{21} & \pi_{22} & \cdots & \pi_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ \pi_{n1} & \pi_{n2} & \cdots & \pi_{nn} \end{pmatrix},$$

(2)

where $\pi_{i,j} = \frac{1}{\sum_{j \neq i} d_{i,j}}$ is the inverse of the Euclidean distance between tasks $\tau_i$ and $\tau_j$ normalized over all tasks.

Note that $\sum_i \pi_{i,j} = 1$. Also, since a robot needs to service a task’s demand at most once, $\pi_{i,i} = 0$. Using this inter-task transition matrix, a robot has a higher likelihood of selecting available tasks that are closer than those that are further away.

The problem facing robot $r_i$ is to select a task $\tau$ using the probabilities in the inter-task transition matrix. However, selecting the probabilities from $M_{\tau}$ does not incorporate the dynamic nature of the system manifested through robots servicing and accomplishing tasks. Therefore, each robot $r_i$ maintains a local copy of $M_{\tau}$ denoted by $M_{r_i,\tau}$, and updates it using its own task servicing information. When a robot services a demand of task $\tau_j$, it sets the column $j$ in $M_{r_i,\tau}$, corresponding to task $\tau_j$ to zero, to indicate that it will not service demands from task $\tau_j$ in the future and consequently, not include $\tau_j$ probability in making its decisions. To select tasks, each robot $r_i$ represents its probability of selecting a task at time $t$ as a vector state matrix $V_{r_i,t}$, which is updated as:

$$V_{r_i}(t) = V_{r_i}(t-1) \times M_{r_i,\tau}(t)$$

(3)
Robot $r_i$’s initial vector state, $V_{r_i}(0)$, is given by:

$$V_{r_i}(0) = (\hat{\pi}_{1i} \hat{\pi}_{2i} \ldots \hat{\pi}_{ni})$$

(4)

where $\hat{\pi}_{ij} = \frac{1}{\sum_{j \neq i} \delta_{ij}}$ is the inverse of the Euclidean distance between robot $r_i$’s initial location and the location of task $\tau_j$, normalized over the distances of $r_i$ to all tasks. Robot $r_i$ makes a decision about the next task to visit and process demand by selecting a next task according to the highest probability of tasks in $V_{r_i}(t)$. Since the robots select tasks in a distributed manner, more than one robot end up selecting the same task. In that case, the task is allocated to the robot with the higher probability of performing the task. If more than one robot have the same probability, the task is allocated to the robot with the highest identifier.

From the probabilities in $V_{r_i}(t)$ a robot can rank a queue of open tasks at time $t$, $Q_{r_i}(t) \subseteq V_{r_i}(t)$, according to the spatial distribution of tasks. The values of $Q_{r_i}$, corresponding to the original values $\pi_{ij}$ sorted in descending order, represent the set of current alternatives for robot $r_i$ at time $t$, as expressed in 5. Note that $q_{1r_i} \in Q_{r_i}$ represents the first option for robot $r_i$.

$$Q_{r_i}(t) = \{q_{1r_i}, q_{2r_i}, \ldots, q_{n(t)r_i} : q_{j+1r_i} \geq q_{jr_i} \land 0 \leq ad_j \leq nd_j\}$$

(5)

### 4.2 Experiments to Compare Proposed MRTA Models

The stochastic queueing based model was compared with the greedy distance based model under identical scenarios of tasks service. In the stochastic approach robots receive a copy of the list of tasks and estimate locally the transition probabilities, vector state and queue of tasks to service. In the greedy approach robots receive the updated list of tasks/demands from a central repository and make locally a decision using this information.

These experiments focus specifically on task allocation, that means that we do not deal with localization or coverage issues in this part of the project. We conducted a set of experiments using simulated robots in the Webots environment. These experiments were performed on an Intel Core i7 CPU 960@3.20 GHz with 12 GB of RAM computer, under 64-bit Windows 7 and using Webots 6.3.0. We used a robot team with 5 robots and 10 different scenarios comprising 18 tasks, each one with a variable number of demands ranging from 3 to 5. The robot model used in these experiments is based on the commercial platform Coroware CoroBot that is shown in Figure 2. We assume a robot is able to reach the point in the environment where a demand of service is active and we assume also reliable communication. In these experiments the stochastic model overcomes the greedy one in all the relevant measurements that were recorded. Figures 6 and 7 contrast for both groups of robots the completion and idle time to achieve a number of demands, and the distance traveled by the robots to service these demands, respectively. Table 3 summarizes the average measurements scored by robot teams using both models to solve similar problems.

From these results we can observe that non only the stochastic model scored better results but also that the variability of the former is smaller compared with the results of the greedy model. Note that the stochastic approach outperforms the greedy one in critical cases where the greedy model produces a bad solution as it is the case of the simulation number 3. Robots using the stochastic model perform additional calculations to make decisions, compared to the ones using the greedy model, however these calculations remain bounded and do not impact in most of the reported cases the time invested to solve a problem.

### 5. COMRADES USER INTERFACE

A graphical user interface (GUI) was programmed to interact with a team of physical robots in a mission of collective detection of landmine-like objects. The primary goal of the GUI is to visualize in real time the state of a mission including the position and current status of the robots, their available battery power, the location of potential landmines detected by robots, etc. A user can also command robots to do specific actions through the
GUI such as to abort its current operation and navigate to a specific location in the environment, recall robots to the base station, and stop and restart the robots.

Figure 8 summarizes the scheme of communications that enable the interaction with the robots through the GUI. Robots communicate asynchronously among themselves and with an external robot server through UDP ports. Each robot sends approximately every 2 seconds data about its pose and the state of its battery. When a landmine-like object is detected the robot sends the estimated position of such object. Landmine-like objects are represented in these experiments by red pieces of paper detectable by the robots’ camera. The communication between the robots’ server and the GUI is established through TCP ports. The robot server updates periodically the state of robots to the GUI, and this part of communication is established synchronously.

| Table 3. Average parameters scored by 6 robots to service 18 tasks in 10 different scenarios |
|----------------------------------------|--------|--------|--------|--------|--------|
|                                        | Greedy Model |                  | Stochastic Model |
|                                        | $\mu$    | $\sigma$ | $\mu$    | $\sigma$ |
| Completion Ticks                       | 15718.40 | 4318.03  | 10887.00 | 2324.14 |
| Idle Ticks                             | 3171.98  | 3039.4   | 1720.18  | 595.68  |
| Exchanged Messages                     | 988.30   | 102.49   | 754.88   | 126.55  |
| Exchanged Bytes                        | 42150.88 | 4590.60  | 31209.7  | 6044.83 |
| Distance                               | 156.53   | 16.02    | 111.83   | 22.46   |
| Battery                                | 426.36   | 9.50     | 447.75   | 10.38   |
Robots can be deployed by the user to specific locations and can also wander around the environment searching for landmine-like objects. Robots are not aware of the position or number of the landmine-like objects in the environment. It is also worth to remark that these experiments concern only autonomous robots that make their own decisions. During the deployment stage, for instance, a robot receives a goal position but it is responsible for planning its path to reach this position.
A repertoire of behaviors such as obstacle avoidance, wandering and mine avoidance was programmed in C++ and replicated in each robot. Each robot is equipped with an onboard computer, an AMD Athlon Dual Core Processor 5050e @2.6Gz with 1.87 GB of RAM computer under Windows XP. The robot server runs in an Intel Xenon CPU E5504@2.0 GHz with 2.00 GB of RAM desktop computer under Windows XP; the GUI was programmed in Java.

Figure 9 shows two snapshots of an experiment involving two physical robots deployed in an indoor experimental environment of $12 \times 17$ square feet and the corresponding state of the GUI. The paths followed by robots are recorded and highlighted, and the landmine-like objects detected by robots during the experiment are represented with red spots in the GUI.

6. DISCUSSIONS AND FUTURE WORK

In this paper we described our research in the COMRADES project for multi-robot landmine detection focussing on multi-robot coverage and multi-robot task allocation. Another important aspect of the COMRADE system is multi-sensor information fusion and we describe a market-based information fusion technique for the COMRADE project in our companion paper. Currently, we are working on implementing the techniques described in this paper on Corobot Explorer robots that are custom-fitted with metal detectors capable of detecting metallic objects like landmines, as shown in Figure 10. We envisage that with the advancement of techniques described in this paper mobile, autonomous robots will mature as a feasible technology for accurately detecting anti-personnel landmines.

REFERENCES