Multirobot/Multiagent Reading Group Notes

Mary Koes

February 26, 2005

1 Motivation

The dangerous nature of a disaster area makes it an ideal application for robotic exploration. Yet the unstructured and uncertain nature of the environment, the necessity for speed and heterogeneous capabilities, and the communication constraints all make it a challenging application for multirobot coordination. Robots may start out with an initial model of the environment, but must be able to adapt their plans as tasks arrive or are removed. Since multiple robots are frequently required to collaborate to accomplish a single task, robots must be able to reason about their commitments to their teammates. Robots must also be able to explain their plans to humans, who may wish to change the plans. A new framework is needed to coordinate in this domain.

2 Problem Definition

Consider a set N robots operating in an environment with K relevant capabilities and a set of M tasks to be accomplished. Each task has an associated reward, location, and duration to accomplish. We then begin with a resource set $\mathcal{R} := \{R^1, R^2, ..., R^N\}$ and a set of tasks or goals $\mathcal{G} := \{G^1, G^2, ..., G^M\}$.

The object is to create a team plan for all the resources to maximize the team reward over the time available.

We assume that the environment is discretized and represented as a weighted undirected graph of nodes and edges, $(\mathcal{N}, \mathcal{E})$.

Each resource R^n has specific capabilities, S^n where S^n is a vector whose length is equal to the cardinality of the set of capabilities relevant to this environment, K. Thus S^{n_k} is 1 if robot n possesses capability k and 0 otherwise.

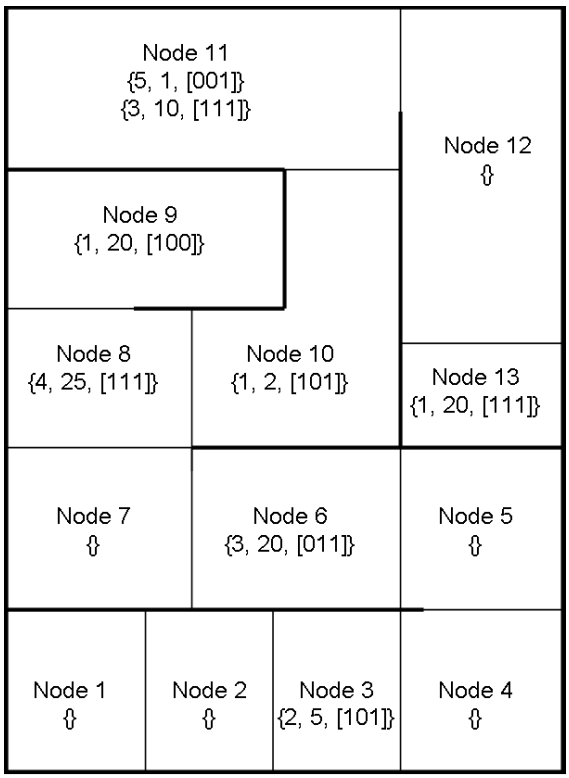

Figure 1: Blueprint used to generate example case. Tasks are represented as {reward, duration, [capability requirements]}. Sample capabilities might include the ability to map the space, check for sound, check for heat, or check for motion.

Each task G^m has an associated location, duration, reward, and necessary set of capabilities. We can then characterize the m-th goal as a tuple $G^m := (N^m, d^m, Q^m(t); C^m)$ where C^m, the capabilities required to achieve goal m, is a vector whose length is equal to K and $C^{m_k} = 1$ if capability k is required to achieve task m and 0 otherwise. The duration requirement means that all capabilities must be satisfied by the resource set, $\mathcal{R}$, at location N^m for the entire continuous duration, d^m, in order for the goal to be achieved. In general, we assume rewards, Q^m,may be time varying.

We assume that there is some time limit T_{max} after which time all rewards are 0. Optionally, we can limit the number of tasks planned for any robot over this duration though in general this value is assumed to be equal to the number of tasks in the problem.

A sample environment is shown in figure 1 and a textual representation is shown in figure 2.

```
Timeslots 8 Maxtime 100 Capabilities 3

#Robot ID
Robot 1 [ 1 0 0 ]
Robot 2 [ 0 1 0 ]
Robot 3 [ 1 0 1 ]

#Task ID, Node,Capability set,Reward,ETC, Include?
Task 3 Node 3 [1 0 1] 5 2 1
Task 6 Node 6 [0 1 1] 20 3 1
Task 8 Node 8 [1 1 1] 25 4 1
Task 9 Node 9 [1 0 0] 20 1 1
Task 10 Node 10 [1 0 1] 2 1 1
Task 11 Node 11 [0 0 1] 1 5 1
Task 111 Node 11 [1 1 1] 10 3 1
Task 13  Node 13 [1 1 1] 20 1 1

#Map
Mapsize 13
Node 1 Node 2 1
Node 2 Node 3 1
Node 3 Node 4 1
Node 4 Node 5 1
Node 5 Node 6 1
Node 6 Node 7 1
Node 7 Node 8 1
Node 8 Node 9 1
Node 8 Node 10 1
Node 10 Node 11 1
Node 11 Node 12 1
Node 12 Node 13 1


#Robot Starting Position:
Start Robot 1 Node 1
Start Robot 2 Node 1
Start Robot 3 Node 11
```

Figure 2: Sample text file for problem input. See figure 1 for environment used to generate this problem.

3 Approach

We want an anytime algorithm that can find a feasible solution for the problem in real time for a small or medium sized group of robots and tasks but that, given enough time, will eventually find the optimal solution. In addition, we want to provide bounds on optimality. Initially, we take a distributed centralized approach, that is that all robots have access to the entire problem and individually compute the solution.

We formulate the problem as a mixed integer linear programming problem (MILP) and use standard linear programming techniques to solve it. In order for the problem to be stated as a MILP, we assume that reward varies linearly with time elapsed. We then define our utility function as the sum of all tasks accomplished discounted by the amount of time it took to accomplish that task

$$Utility = \sum_{m \in \mathcal{G}} \frac{T_{max} - G_m\_start}{T_{max}} Q^m$$

where $G_m\_start = T_{max}$ if goal m is not accomplished and G_m_start equals the time at which the task is started if the task is accomplished.

Since MILP solvers are typically not adept at solving long sequential plans and can be much slower at path planning than A* or other search methods, we attempt to reduce the path planning complexity by transforming the original representation into a goal-oriented representation. Robots then create a plan of task orderings (figure 3). For example, the resulting solution may be for robot 1 to work on task 3 first and then work on task 2 and for robot 2 to work on task 2 first and then do task 8. However, as before, robots must be colocated in both time in space in order to work on a task together. Each subplan consists of the time to travel to the location to perform the task, the time spent waiting for its teammates to arrive, and the time to actually accomplish the task (figure 3).

The transformation from the original environment $(\mathcal{N}, \mathcal{E})$ into a new fully connected graph $(\mathcal{N}_G, \mathcal{E}_G)$ where each task $G^m \in \mathcal{G}$ is mapped to a node in $\mathcal{N}_G$ requires that path planning be precomputed (we use Floyd's algorithm which is $O(n^3)$). In general, the edge costs may differ for different classes of robots. Multiple goals at a single node in $\mathcal{N}$ will be modeled as multiple nodes in $\mathcal{N}_G$ with edge lengths of 0 between them.

In addition to the task nodes, we also have a start node and an idle node (Figure 4). The start node has directed edges to each task node in the graph as well as the idle node and represents the starting positions of the robots. The edge costs of the start node edges are likewise computed from $(\mathcal{N}, \mathcal{E})$. If the robots start at different nodes in $(\mathcal{N}, \mathcal{E})$, this may be modeled either by allowing multiple start nodes or having heterogeneous costs to different task nodes. Every node has an edge with zero cost to the idle node. The idle node has no

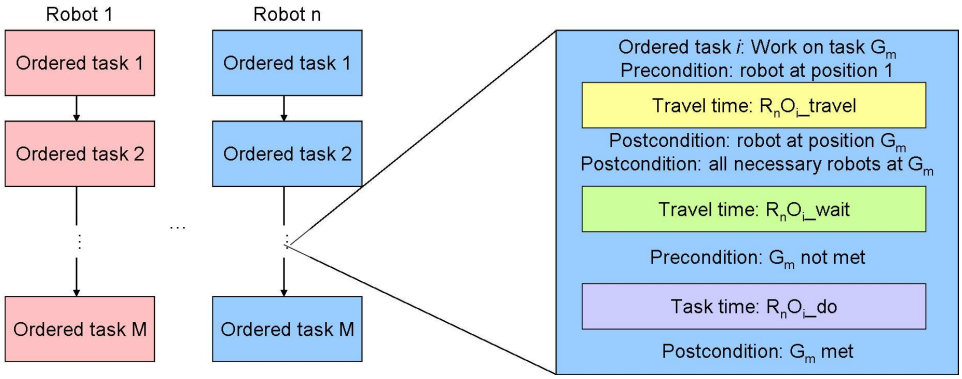

Figure 3: Sample solution for proposed problem formulation–an ordering of tasks to work on for each robot.

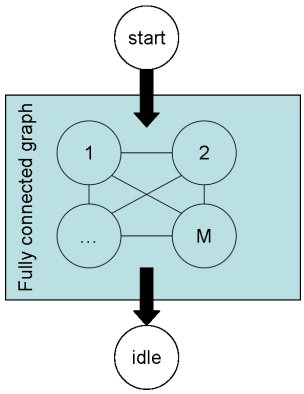

Figure 4: $(\mathcal{N}_G, \mathcal{E}_G)$ showing relationships between task nodes, start, and idle nodes

transitions out since a robot would never schedule a timeslot as idle unless there were no more possible tasks for it to work on.

4 Related Work

Disclaimer: The following list is in no way complete. If you're interested for references for any of these topics, please let me know.

- Operations Research: Obviously, work in MILP is relevant to this approach. We use CPLEX as our MILP solver.

Much research has been done on constraint reasoning and, recently, distributed constraint reasoning. One notable algorithm is Adopt, designed for distributed asynchronous constraint optimization. We attempted to apply ADOPT to a simplified USAR problem (ignoring all spatial constraints) and found that it was prohibitively slow, primarily because it cannot leverage the relaxed solution as a MILP solver can and must use only branch and bound. Its performance is further compromised by its polynomial space complexity guarantees. Rather than storing the space tree, it must sometimes search solutions multiple times. Both in terms of total solution time and total communication, a distributed centralized approach outperformed ADOPT.

- Multicommodity dynamic flows: Dynamic flows, or flows over time, model problems where travel is not instantaneous. They have traditionally been modeled as time-expanded networks by copying the original network at each time step. The enormous size of the time-expanded network is a major problem (we originally used a time-expanded network to represent this problem but quickly found the solutions infeasible for large distances). Hall, Hippler, and Skutella demonstrated that, without costs, multicommodity flows over time are NP-hard.

- Market based coordination: we want to explicitly maximize *team* reward rather than individual reward. We also want to provide optimality bounds guarantees.

- Other?

5 Planning on replanning

One obvious problem with this approach is that the USAR environment is uncertain so the initial model of the environment will almost certainly be wrong. Tasks may take more or less time than expected. New tasks will be added or old tasks removed. Additional robots may arrive or existing robots may get stuck. Estimates of path lengths will change. Rather than explicitly model this uncertainty, we simply use the expected value and then plan on replanning.

A decision theoretic framework is needed to decide what to do when an agent receives information indicating that the information on which the current plan is based is wrong. Since (at least when the information is synchronized) the agent knows the plans of all its teammates, it can decide on whether replanning is necessary, whether the information should be shared, and what the strategy for replanning should be.