Towards Replanning for Mobile Service Robots with Shared Information

Brian Coltin and Manuela Veloso

School of Computer Science, Carnegie Mellon University
500 Forbes Avenue, Pittsburgh, PA, 15213
{bcoltin,veloso}@cs.cmu.edu

Abstract. Currently, multiple mobile robots, called CoBots, are deployed in an office building performing tasks for users such as delivering items and escorting visitors. Tasks for these robots are scheduled in parallel, but aside from the scheduling, the robots act entirely independently. In this work we explore how the robots can communicate dynamic changes in the environment, such as blocked corridors, closed doors at delivery locations, and robot failures, to accomplish their tasks more effectively. We introduce the concept of a "rationale graph" which documents the relationships between dynamic conditions and scheduling decisions. We propose to use the rationale graph to decide which information to share with which robots, and to replan for tasks which have rationales that are violated.

Keywords: multi-robot, planning, replanning, information sharing

1 Introduction

As mobile robots become increasingly capable of moving in the environment autonomously, multi-agent mobile robot systems become more and more plausible. In fact, we already have four CoBot robots that navigate in an office building, fulfilling requests for users such as delivering messgaes, delivering items, attending meetings with telepresence, and escorting visitors.

The robots are scheduled to perform multiple tasks at the same time. When picking up and delivering items, they transfer items between each other to accomplish their tasks more efficiently. However, aside from these two factors, the robots act completely independently and accomplish their tasks on their own. It is a multi-agent system, but there is little to no close cooperation between the robots.

In this paper, we seek to change the lack of close multi-agent cooperation in our service robots. We are not interested in having the robots perform new tasks in which multi-agent techniques are more immediately applicable, but in making use of the fact that there are multiple robots to extend and improve the robots' existing capabilities.

Particularly, we look at novel ways to improve task execution through distributed information sharing. Robots can detect when doors where deliveries are requested are closed, and when corridors are crowded or blocked. We look at constructing a rationale graph detailing the dependencies between these assumptions and the robots plan. We then use this rationale graph to communicate which assumptions have failed and to replan for the affected tasks on the rationale graph. Replanning can be done in both centralized and decentralized fashions.

Many replanning schemes have been proposed for dynamic task allocation problems [1, 2]. Fewer have looked at replanning schedules with a sequence of timed, spatially located tasks. One approach has been to iteratively repair schedules in a centralized manner [13]. Another group has looked at maintaining Multi-agent Simple Temporal Networks in a distributed setting to maintain time constraints [**?**].

In the remainder of this paper, we present the CoBot robots and their capabilities, and detail the problem being solved. Then, we discuss rationale graphs, how broken rationales can be communicated among robots, and how the robots can replan in response to unexpected events.

2 Executing Tasks on the CoBots

We have developed the CoBot robots, which navigate autonomously in an office building performing tasks for users. There are currenlty four CoBots, three of which are deployed in the same building (see Figure 1) and one which is deployed off-site. Each CoBot has an omnidirectional base, a scaled up version of our small-size RoboCup soccer robots, designed by Mike Licitra. They have a Kinect sensor to localize and detect obstacles, a touchscreen to interact with humans, and microphones and speakers for voice interactions. Cobot1 and CoBot2 have a LIDAR sensor for localization. In CoBot3 and CoBot4 the LIDAR has been replaced with an additional inexpensive Kinect sensor.

To fulfill these tasks, CoBot localizes in the building using Corrective Gradient Refinement localization on a vector map [6]. It navigates on a graph of the environment, and autonomously avoids obstacles. Since CoBot does not have arms to manipulate the world with, it relies on human help for some of its capabilities, utlizing the idea of symbiotic autonomy, where robots and humans cooperate to accomplish tasks. CoBot asks humans to place items in its basket and to help it press elevator buttons [7]. If necessary, CoBot seeks human help proactively, visiting offices to ask for help and sending an email to our research group if no help can be found [8]. CoBot has completed hundreds of tasks and travelled hundreds of kilometers to do so [7].

3 Scheduling Tasks for the CoBots

The CoBots are deployed to complete user tasks with little supervision. Users request a task on a website, and specify a window of time that the task should be completed in.

The tasks users are able to schedule include:

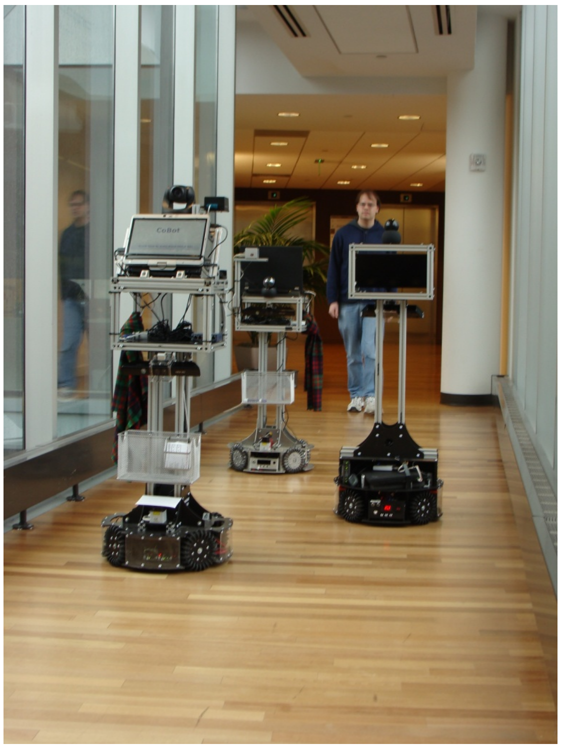

Fig. 1. CoBot-1, CoBot-2, and CoBot-4 are deployed in the Gates Hillman Center to fulfill user tasks.

- **Visit a room:** CoBot goes to a room to introduce itself.
- **Deliver a Spoken Message:** CoBot travels to a room and speaks a message entered by the task requester to a room's occupant.
- **Escort a Visitor:** CoBot meets a visitor at the elevator and leads them to a room.
- **Pick Up and Delivery:** CoBot goes to one room, asks a user to place a specified item in its basket, and delivers that item to a different room. Users have used this tasks to send various items including printouts, paper revisions, USB keys, money owed for lunch, mail received by secretaries, and snacks.
- **Telepresence:** CoBot goes to a room and makes its telepresence interface available for trusted users to attend a meeting. Users can control CoBot over the web and communicate via videoconferencing software [4].
- **Multi-Object Delivery:** Retrieve a set of items from a single location and deliver to multiple locations. Examples include delivering candy and mail.
- **Multi-Object Retrieval:** Retrieve a set of items from multiple locations and deliver to a single locations. Examples include picking up mail from secretaries and delivering it to a central office, delivering coffee and returning to

a central location with payments, or delivering business cards and returning the leftovers.

- **Find and Retrieve an Object:** The user tells CoBot to find an item by name, such as "towel", "pen", or "coffee", and the robot queries OpenEval [5] to decide whether to go to an office, kitchen, common area, or restroom.

Each task T_i must be assigned:

- A start time t_i such that, given a window of time $[s_i, e_i]$ and an expected duration δ_i, $s_i \leq t_i + \delta_i \leq e_i$, and
- A robot r_i to execute the task such that, given a set of valid robots $V_i \subseteq R$ able to execute the task, $r_i \in V_i$.

Furthermore, no two tasks may overlap, and given an estimated time to travel between two locations $d(A, B)$;

- For all task pairs T_i, T_j such that $r_i = r_j$, which occur at locations l_i and l_j, either $t_i + \delta_i + d(l_i, l_j) \leq t_j$, or $t_j + \delta_j + d(l_j, l_i) \leq t_i$,
- For two tasks T_i and T_j that have a precedence constraint (such as a pickup task and a delivery task), $t_i < t_j$.

The objective is either to complete all tasks as early as possible ($\min \sum t_i$) or to minimize the distance travelled by all of the robots, conserving fuel. Each task is assigned a robot and execution time by a centralized scheduler which solves a mixed integer programming problem [3]. Additional tasks can be scheduled online by speaking with the robot. The scheduler chooses the ordering of the tasks and their assignment to robots, while a separate path planner onboard the robot chooses which path the robot will take to go from place to place.

For example, let's assume six tasks are requested:

- A pickup task T_1 and a delivery task T_2 to pickup a paper with revisions from room 7205 and deliver it to room 7127. For both tasks, $s_1 = s_2 =$ 3:00 PM and $e_1 = e_2 =$ 3:30 PM. $V_1 = V_2 = R = \{$ CoBot-1, CoBot-2$\}$.
- A task T_3 to deliver a message to the occupant of office 7110, with $s_3 =$ 3:10 PM, $e_3 =$ 3:20 PM, and $V_3 = R$.
- A task T_4 to pick up a visitor from the elevator and a task T_5 to deliver the visitor to room room 7213, with $s_4 = s_5 =$ 3:00 PM, $e_4 = e_5 =$ 3:10 PM, and $V_4 = R$.
- A task T_6 to deliver a message to the occupant of office 7801, with $s_6 =$ 3:10 PM, $e_6 =$ 3:15 PM, and $V_6 = R$.

The task locations and the robots' initial positions are shown in Figure 2. One potential schedule would be:

CoBot-1		CoBot-2	
Time	Task	Time	Task
3:00 PM	T_1 : Pick up paper.	3:00 PM	T_4: Meet visitor at elevator.
3:10 PM	T_3 : Deliver message.	3:07 PM	T_5: Deliver visitor.
3:15 PM	T_2 : Deliver paper.	3:12 PM	T_6: Deliver message.

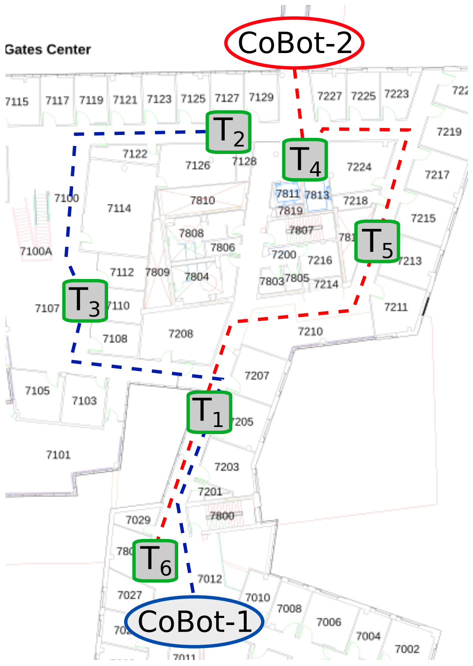

Fig. 2. Robot starting positions, task locations, and the generated plan.

The tasks are divided between the robots to complete more tasks more quickly.

For *Multi-Object Delivery* and *Multi-Object Retrieval* tasks, robots collaborate by *transferring* items to make the deliveries more quickly and at lower cost [9]. We are currently researching algorithms to take advantage of transfers when performing *Pick Up and Delivery* tasks.

However, aside from forming multi-robot schedules which may include transfers, we take little advantage of the fact that there are multiple robots deployed in the building. Once assigned a task, a robot executes that task independently without aid or communication from any other robots. Multi-agent techniques have obvious uses for other tasks that could be performed on the robots, such as patrolling, exploration and coverage. In the remainder of this paper, we explore how multi-agent techinques can be applied to the CoBots to perform their existing tasks more effectively.

4 Multi-Agent Replanning with Rationale Graphs

One of the main factors preventing service robots from being deployed in the real world is a lack of robustness to a dynamic world with unexpected events. This robustness can be increased through the use of multiple agents. Specifically, we propose to increase robustness among the CoBots by sharing information and using this information to replan. The information we envision the CoBots sharing includes:

– *Hallways that are blocked or have heavy traffic.* If one robot cannot navigate through a hallway, other robots should avoid that corridor until traffic decreases.
– *Closed and open doors.* Robots have some flexibility regarding when they deliver a message or item to an office. If one robot passes an office and sees it is currenlty closed, other robots should delay making trips to that office, if their schedule constraints allow it.
– *Robot failures.* If a robot is unable to move or the other robots are unable to communicate with it for a lengthy period of time, it will not be able to complete its tasks. The other robots should finish the tasks in its stead.
– *Human help availability.* CoBots rely on humans to help them take the elevator. If one robot sees humans waiting at the elevator, it could communicate this information to let the other robots know that it may be a good time to take the elevator. Also, communicating open door information can help other robots proactively seek help in occupied offices.

This shared information serves as preconditions in each robot's schedule. To complete a delivery task, the hallways the robot travels in must not be blocked, the door at the delivery point must be open, the robot must be in working order, and human help must be available when necessary. Each piece of information is time-sensitive, and the robots' certainty decays over time.

We assume that the multi-agent planner outputs these prerequisites, or *rationales* for the plan along with the plan itself. For example, a plan to deliver two items to Office A and Office B could be:

1. $goto(r_1, A)$ because $open(A) \wedge unblocked(\text{Hall1})$
2. $goto(r_1, B)$ because $open(B) \wedge unblocked(\text{Hall2})$

If an assumption behind the plan's reasoning changed, such as another robot observing Hall1 is blocked, then replanning would become necessary. The planner could adjust the plan so that the robot went down a different hallway, or so that a different robot which could take a different path performed the task instead.

The largest difficulty is that the planner must provide reasoning along with a plan. Sometimes the reasoning behind a plan is not entirely clear. Negative reasoning is particularly difficult to come up with: if a robot chose to take one corridor first, there could be any number of other corridors it could have chosen had they been less crowded, or any number of other tasks that could have been completed earlier had one corridor had a different state or one door been open earlier.

However, positive rationales are very easy to discover. In the earlier example, office A was chosen first because it was assumed to be open, and Hall1 was selected as part of the robot's path because it was unblocked. If these rationale are violated then the plan is highly likely to change. If another path becomes open or a door opens, then a better plan could become available, but the current plan is still valid. So we focus on positive justifications. If negative justifications are returned by the planner, they can still be shared and exploited as well.

Reasoning could be used to replan for even greater effect if a *rationale graph* can be formed. Here, each choice predicate (or a subset of the predicates) is linked to the specific decisions in the plan which came about as a result of that predicate. Each of these decision points / scheduling assignments links to another choices. If this graph is known, then when a predicate changes, only the parts of the plan that are its dependences in the rationale graph need to be updated. A rationale graph will likely be incomplete, with only positive rationales or with only selected negative rationales, since the set of negative justifications may be large.

When a prerequisite is violated (i.e., a hallway is blocked), the tasks linked to it are rescheduled in light of the violated assumption. For example, in Figure 3, CoBot-2 is heading towards an office to make a delivery. However, CoBot-4, while heading down the same hallway in the opposite direction, encounters an obstacle and is blocked. It proceeds to communicate this information to CoBot-2, which replans and selects an alternate route to its destination which avoids the blocked hallway. The original and revised routes are shown in Figure 4.

As a second example, in Figure 5, CoBot-2 is heading to Office A to deliver an object. CoBot-4, in an unrelated task, passes by Office A and observes that the door is closed. It communicates this information to CoBot-2, which proceeds to head to Office B to complete a separate task, as it has some leeway in its deadlines. After accomplishing its task, CoBot-4 heads back past office A and sees that the door is now open. CoBot-2 can now head to Office A and complete its delivery.

Two major multi-agent technical challenges must be solved to accomplish this behiavor. First, the robots must have a method of sharing prerequisites in the rationale graph and deciding which information to share with one another. Second, the robots must have an algorithm to adjust a multi-agent plan and decide when to replan.

5 Multi-Agent Rationale Sharing

The first challenge to replanning with shared information is choosing how to share rationales and selecting which rationales to share. We present two potential mechanisms for information sharing.

5.1 Distributed World Model

In the first method, a distributed world model, all robots share complete information of the world. This is a common technique in multi-agent robot soccer [10, 11]. At regular intervals, each robot shares its full state with all other robots: it's position and orientation, its current plan, and the observed state of the world, including open and closed doors and blocked hallways. The robots merge these observations to store the current state of each robot, the state of each door and

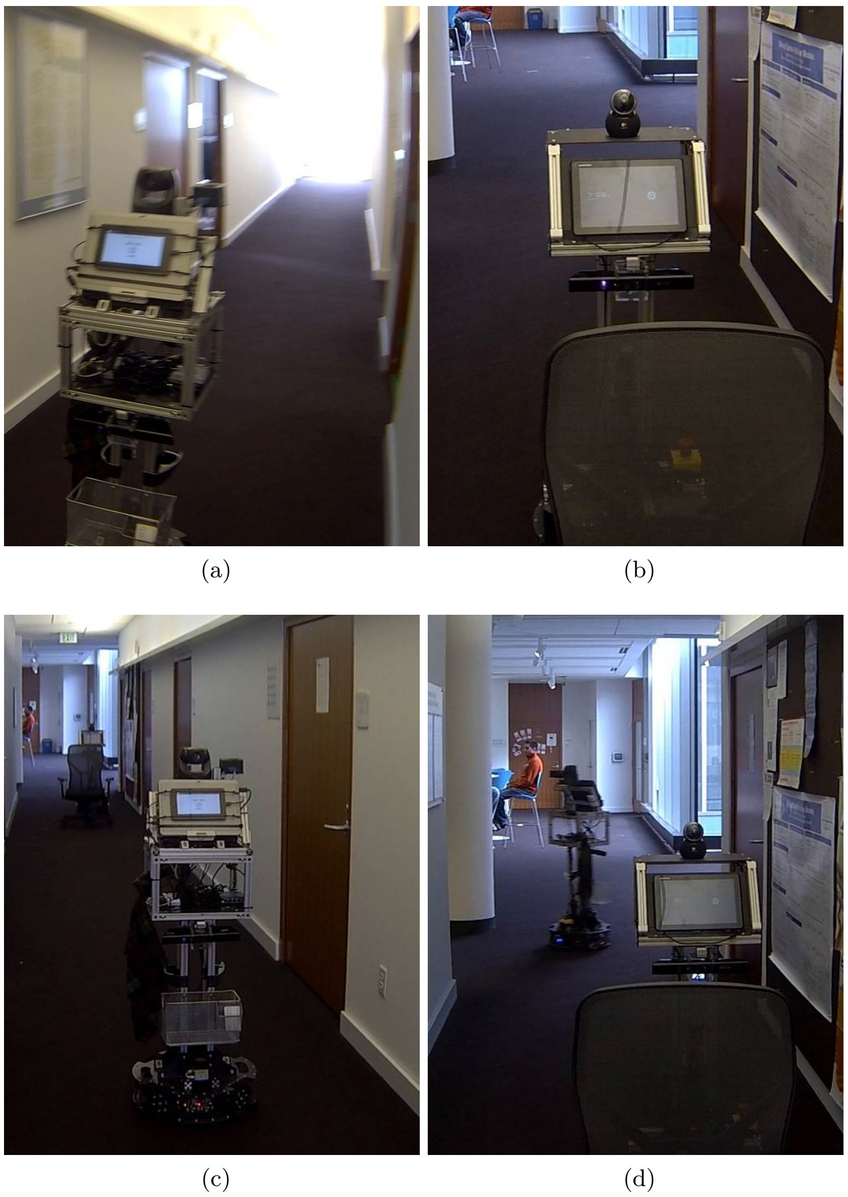

Fig. 3. (a) CoBot-2 heads towards an office to make a delivery. (b) CoBot-4 detects that a hallway is blocked, and communicates this information to CoBot-2. (c) CoBot-2 uses the information to change its plan. (d) CoBot-2 takes an alternate round to avoid the blocked hallway.

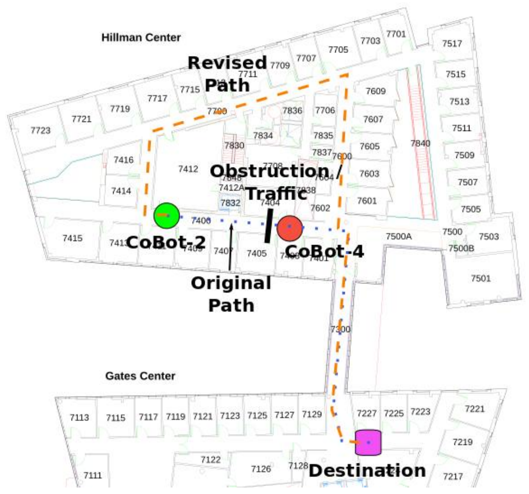

Fig. 4. CoBot-2 replans its route after CoBot-4 reports that a hallway is blocked.

the last time it was observed, and the state of each hallway and the last time it was observed. Hallway observations could consist of the last time a hallway was traversed and the time it took, or a "blocked" status if a robot is currently unable to progress in a hallway.

Forming a distributed world model makes it easy for each robot to have all necessary information to detect violated rationales in the rationale graph, and to plan and to replan. It is also easy to detect when a robot has failed from a lack of communication. For a relatively small deployment a distributed world model may be the best option. However, it does not scale well to large numbers of robots, where each robot needs to share its entire state with n other robots. Furthermore it requires significant bandwidth use, which is already limited by other applications such as telepresence which require a large part of the available wireless bandwidth.

5.2 Distributed Sharing of Invalidated Rationales

In the second method, the entire rationale graph is shared with all of the robots. When a robot detects an invalidated rationale, it shares the information only with the robots assigned tasks directly linked to that prerequisite in the rationale graphe.

This approach requires little bandwidth compared to the frequent communication updates needed to form a distributed world model. Furthermore, it is less susceptible to network delays and failures, which are common with mobile robots required to switch between multiple wireless access points. The planner could be centralized, like our current MIP approach, or tasks could be assigned in a decentralized, online fashion.

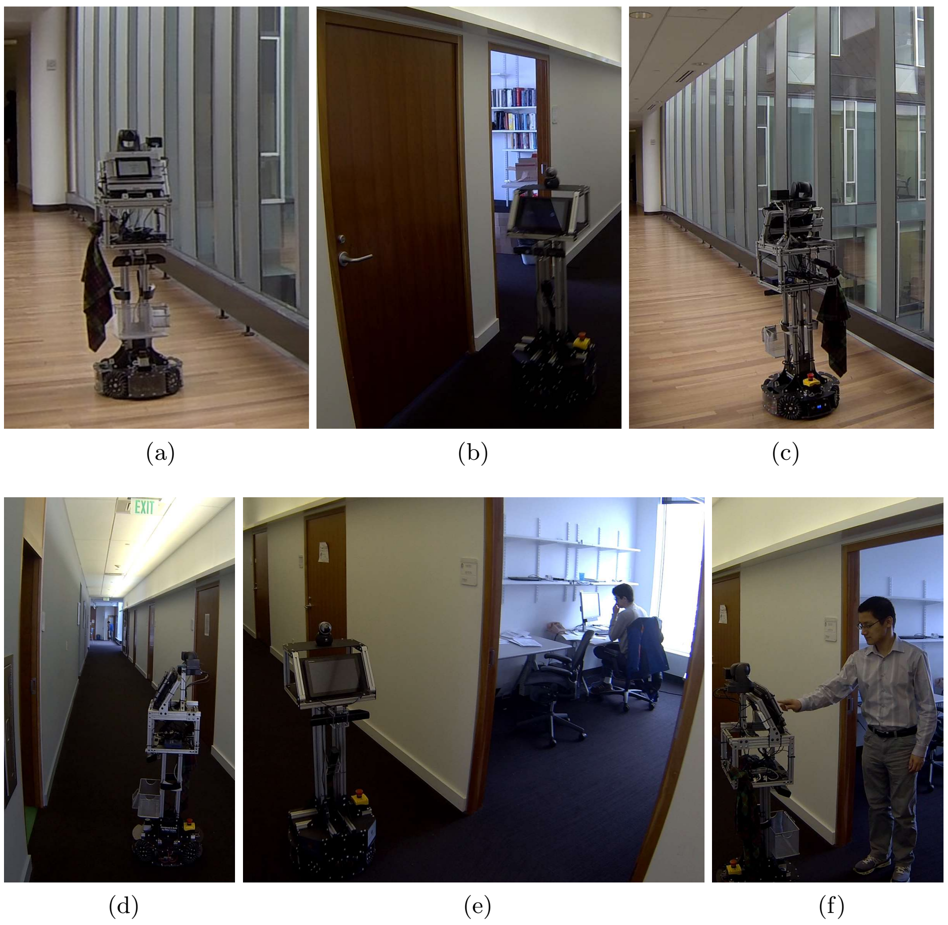

Fig. 5. (a) CoBot-2 heads to Office A to make a delivery. (b) CoBot-4 passes Office A and observes the door is closed. (c) CoBot-2 replans and decides to complete a task at Office B first, turning around. (d) CoBot-2 completes its task at Office B. (e) CoBot-4 returns from its previous task and observes that the door to Office A is now open. (f) CoBot-2 completes its original task at Office A.

However, a separate method is required to determine when a robot fails. This could be accomplished by scheduling regular communicated updates with specific robots. If an update does not occur, the robot can be assumed to have failed.

6 Multi-Agent Replanning

The remaining technical difficulty is how the robotic agents can replan given shared information, either from a distributed world model or from more limited reason-based information sharing. We discuss two potential methods of replanning: centralized replanning, and decentralized replanning, particularly by auctioning tasks to robots.

6.1 Centralized Multi-Agent Replanning

Currently a schedule is formed on a centralized survey and then sent to the robots to execute. This setup makes sense since there is already a centralized server to accept user tasks on the website. The schedule is currently formed by solving a mixed integer program. However, for the larger problem instances presented by the *Multi-Object Retrieval* and *Multi-Object Delivery* tasks, we have developed a much faster heuristic approximation algorithm that uses transfers [9]. Other researchers have developed many other exact and approximate methods to schedule solutions to vehicle routing problems and pickup and delivery problems such as ours [12].

The simplest way to replan is, when one of the original plan's rationales is canceled, to form a new plan entirely from scratch. However, this is computationally expensive and is often unnecessary. Furthermore, it may sow confusion among users if the users were given a specific time a robot is expected to come: for this reason, we only tell users that a CoBot will arrive in their requested time window. Other researchers have studied how to adjust schedules while minimally impacting existing schedules [13].

Instead of planning entirely from scratch, we can can make use of the rationale graph to partially replan only the tasks on the schedule whose rationales are broken. This can be done with a local search procedure, such as Tabu search or simulated annealing, searching for small changes to the affected area to find a new schedule. The rest of the plan that is not affected by the rationale graph is treated as a base and remains unchanged.

If a full rationale graph is not available from the planner, then local search can be performed on the entire plan with an additional weight towards changing the links causing direct conflicts.

When replanning, care must be taken so that the schedule includes the current state of the robot: the tasks that robots are already in the progress of completing, including pickup tasks the robots have already completed and items they are currently carrying.

6.2 Multi-Agent Replanning with Decentralized Auctions

Planning and replanning of multi-agent schedules can also be done in a distributed, online manner using auctions. Auction mechanisms have previously been used for other vehicle routing problems, such as scheduling passengers for ridesharing [14, 15]. When one agent receives a request (it could be either the web server or a user communicating with a robot directly) that robot broadcasts the request to the others to declare an auction. Each robot then checks if the new request could be fit into its schedule. If so, each robot places a bid with the cost in fuel and time of adding the request into its schedule. The bid with the lowest cost wins, and the auctioneer announces the winner to all robots.

Then, the winner communicates the positive rationales for its bid to the other robots. Each robot stores all the other robots assigned tasks along with their rationales. If a rationale is violated, the relevant robot is informed. That robot then attempts to adjust its own schedule using local search to still complete the task. For some violated rationales no rescheduling is necessary, for example, if a hallway is blocked and the path planner can select a different route. If the task cannot be completed, or the cost is significantly higher, the robot declares a new auction for the same task, to see if another robot can perform it more efficiently.

The rationale graph in this setup does not include negative rationales. However, negative rationales may be included at an additional communication cost. When placing bids, robots can share all the rationales for their bids, both positive and negative. If a negative rationale changes, and the robot that requested the task can now perform it at lower cost, it can communicate with the bid winner to initiate a new auction.

While sharing either positive rationales or all rationales, edges between tasks assignments on the rationale graph do not need to be shared. This is because they can be represented implicitly with the auction mechanism. When a rationale is violated and a new auction is held for a task, the two robots whose schedules changed may reexamine each of their tasks and bids. If one of their tasks comes at a much higher cost now, they may reinitiate the auction. Likewise, if a bid would have been higher (or been possible to make at all) given the new schedule, they may request that the task's current executor initiate a new auction to replan. In this manner, the auction mechanism recurisvely replans for intra-task links in the rationale graph.

7 Conclusion

In this work, we proposed and discussed centralized and distributed methods for replanning from shared rationales, represented as a rationale graph. We discussed methods of efficiently sharing important information between robots on a need to know basis in the form of a rationale graph, and how to apply the rationale graph to both centralized replanning and distributed auctions. Much future work remains, including implementing these algorithms on robots and conducting experiments to validate them. Furthermore, additional research is needed in how to form rationale graphs while planning and selecting which information to include.

Acknowledgements

This research was partially sponsored by the Office of Naval Research under grant number N00014-09-1-1031, and by the National Science Foundation under award number NSF IIS-1012733. The views and conclusions expressed are those of the authors only.

References

1. Parker, L.: L-alliance: Task-oriented multi-robot learning in behavior-based systems. Advanced Robotics **11**(4) (1996) 305–322
2. Botelho, S., Alami, R.: M+: a scheme for multi-robot cooperation through negotiated task allocation and achievement. In: Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on. Volume 2., IEEE (1999) 1234–1239
3. Coltin, B., Veloso, M., Ventura, R.: Dynamic user task scheduling for mobile robots. In: Proc. of the Work. on Automated Action Planning for Autonomous Mobile Robots, AAAI. (2011)
4. Coltin, B., Biswas, J., Pomerleau, D., Veloso, M.: Effective semi-autonomous telepresence. RoboCup 2011: Robot Soccer World Cup XV (2012) 365–376
5. Samadi, M., Kollar, T., Veloso, M.: Using the web to interactively learn to find objects. In: Proceedings of the Twenty-Sixth Conference on Artificial Intelligence (AAAI-12), Toronto, Canada. (2012)
6. Biswas, J., Coltin, B., Veloso, M.: Corrective gradient refinement for mobile robot localization. In: Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on, IEEE (2011) 73–78
7. Veloso, M., Biswas, J., Coltin, B., Rosenthal, S., Brandao, S., Merili, T., Ventura, R.: Symbiotic-autonomous service robots for user-requested tasks in a multi-floor building. In: Proc. on IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). (2010) 2932–2937
8. Rosenthal, S., Veloso, M.: Mobile robot planning to seek help with spatially-situated tasks. In: Twenty-Sixth AAAI Conference on Artificial Intelligence. (2012)
9. Coltin, B., Veloso, M.: Optimizing for transfers in a multi-vehicle collection and delivery problem. In: Proc. of DARS. (2012)
10. Coltin, B., Liemhetcharat, S., Meriçli, C., Tay, J., Veloso, M.: Multi-humanoid world modeling in standard platform robot soccer. In: Humanoid Robots (Humanoids), 2010 10th IEEE-RAS International Conference on, IEEE (2010) 424–429
11. Kaminka, G., Frenkel, I.: Integration of coordination mechanisms in the bite multi-robot architecture. In: Robotics and Automation, 2007 IEEE International Conference on, IEEE (2007) 2859–2866
12. Parragh, S., Doerner, K., Hartl, R.: A survey on pickup and delivery problems. Journal für Betriebswirtschaft **58**(2) (2008) 81–117
13. Rubinstein, Z., Smith, S., Barbulescu, L.: Incremental management of oversubscribed vehicle schedules in dynamic dial-a-ride problems. In: Twenty-Sixth AAAI Conference on Artificial Intelligence. (2012)

14. Coltin, B., Veloso, M.: Towards ridesharing with passenger transfers. In: Proc. of AAMAS. (2013)
15. Kleiner, A., Nebel, B., Ziparo, V.: A mechanism for dynamic ride sharing based on parallel auctions. In: 22th International Joint Conference on Artificial Intelligence (IJCAI). (2011) 266–272