On Scaling Multi-Agent Task Reallocation Using Market-Based Approach [*]

Rajesh K. Karmani Timo Latvala Gul Agha
Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, IL 61801 USA
{rkumar8,tlatvala,agha}@uiuc.edu

Abstract

Multi-agent systems (MAS) provide a promising technology for addressing problems such as search and rescue missions, mine sweeping, and surveillance. These problems are a form of the computationally intractable Multi-Depot Traveling Salesman Problem *(MDTSP). We propose a novel market-based approach, called Market-based Approach with Look-ahead Agents (MALA), to address the problem. In MALA, agents use look ahead to optimize their behavior. Each agent plans a preferred, reward-maximizing tour for itself using our proposed algorithm which is based on the Universal TSP algorithm. The agent then uses the preferred tour to evaluate potential trades with other agents in linear time – a necessary prerequisite for scalability of market-based approach. We use simulations in a two dimensional world to study the performance of MALA and compare it with* O-contracts *and* TraderBots*, respectively, a centralized approach and a distributed approach. Experiments suggest that MALA efficiently scales to thousands of tasks and hundreds of agents in terms of both computation and communication complexity, while delivering relatively good-quality but approximate solutions.*

1. Introduction

Multi-agent systems have been proposed as a promising technology to address problems such as *search and rescue missions*, *sensing and data collection*, *mine sweeping*, *surveillance*, etc. We motivate the problem with two examples. Consider a team of robots is dropped from a helicopter for mine-sweeping. The robots know the suspected locations of the mines. The mission of the team is to visit the sites, and detect and disable mines that are found. Each robot has limited battery power. The problem is to find an allocation of mines to robots so that each mine gets visited once by some agent and the mission is accomplished in the shortest time feasible. As another example, consider millibots [10] launched from different points in order to monitor the structural health of a building. For such monitoring, vibration and strain need to be measured at hundreds and thousands of points of interest.

One way to address the problem described above is by using a group of robotic agents to visit a set of pre-selected observation points and accomplish an objective such as taking a measurement. We'll call such a 'visit' to an observation point, a *task*. We assume that we are working with a 'known' world: each agent knows the position corresponding to the tasks but need not know the position of other agents.

An agent's *mission* consists of a set of tasks that are assigned to it, together with the agent's plan to visit these tasks (a tour). A mission is complete when every task in the set has been performed, i.e., the tour is complete. Observe that computing optimal mission plans corresponds to solving the *Multi-Depot Traveling Salesman Problem* (MDTSP), and is thus a generalization of the classic *Traveling Salesman Problem* (TSP). The MDTSP problem is computationally intractable; it involves computing multi-agent TSP solutions. A naïve solution would be to exhaustively enumerate the possible allocations and compute a tour for each allocation. However, this approach is not computationally feasible: given m agents and n tasks, there are m^n allocations. Each allocation requires computing m NP-Hard TSPs, where the size of a TSP varies from 0 to n.

To systematically tackle the problem, we model our agents in the following way. First, each agent is rational but has bounded computational power. Second, an agent is self-interested but cooperative: it wants to maximize the number of tasks that it performs. Finally, each agent is capable of covering only a fixed total distance (corresponding to its fuel). Given this constraint, the global objective becomes that of servicing maximum number of tasks.

We propose a distributed, market-based approach to find

[*]This research has been supported in part by NSF under grant CNS 05-09321 and by ONR under DoD MURI award N0014-02-1-0715.

reasonably efficient, but not optimal, solutions to the task allocation problem. Market-based approaches have been successfully used to solve co-ordination problems for multi-agent systems [1, 5, 8, 17]. They naturally form a distributed framework, and offer a good way of sharing information between agents. The challenge faced by economic approaches, including market-based ones, is to design *efficient* algorithms for the behavior of individual agents.

Our approach *Market-based Approach with Look-ahead Agents* (MALA) works as follows. We start the process by randomly allocating each task to some agent. The problem then becomes one of task reallocation. Each agent then computes a tour on the set of tasks within its fuel range using a modified *Universal TSP (U-TSP)* algorithm [9]. Specifically, we use a 2-approximation TSP tour [3] to guide the non-deterministic choices that are inherent in the original U-TSP algorithm. Our simulations suggest that the use of 2-approximation TSP tour significantly reduces the cost of tours obtained using the original U-TSP algorithm. Note that the 2-approximation tour is only used for resolving the non-determinism, and this does not modify the properties of a U-TSP tour.

The U-TSP based algorithm is desirable due to two reasons. First, observe that some tours computed over the tasks within the fuel range may still exceed the given fuel budget. We propose a greedy pruning strategy which exploits the representation and properties of U-TSP tour, and allows agents to compute their desired tour for a given fuel budget. Second, a U-TSP tour allows linear time computation of the cost of subtours while guaranteeing a good approximation on the tour cost.

The desired tour approximates the maximization of agents' reward for given fuel constraint. Once an agent has computed a desired tour, it negotiates with other agents to get as close as possible to their desired tour. Myopic agents run the risk of getting stuck on a local optimum. Performing a full look ahead in the future would address this problem. In an agent-based decentralized solution, such a look ahead requires each agent to estimate other agents' preferences. This is prohibitively expensive for large instances [12]. We conjecture that a partial look ahead reduces the chances that the solution gets stuck in a local optimum.

We evaluate MALA by means of simulations and compare it with two existing approaches specifically the centralized O-contracts approach [11] and the distributed, multi-agent Traderbots approach [7] . Results suggest that MALA outperforms the centralized approach in solution quality as well as computation time, while the Traderbots approach suffers from higher communication cost.

2. Related Work

Different approaches such as linear programming, mixed integer linear programming and iterative network flow have been adopted to overcome the computational challenge in solving MDTSP. In this paper, we will compare our approach to other market-based approaches–not only because such approaches are the most closely related to our approach, but also because they show considerable improvement in solution quality over the other approaches.

One approach that has been proposed to avoid the intractability of full look ahead may be termed the *instantaneous assignment* (IA) (e.g. [1]). In the IA approach, agents do not plan for tasks beyond the immediate one; instead they focus on the one task they are carrying out or they are bidding for. Predictably, IA can lead to poor solution quality due to local optima.

Sujit et al. [15] restrict the scope of the problem by assuming that the field is unknown, and hence the agents cannot plan ahead. In their work, agents can communicate within a limited range (global broadcast is not feasible). Their approach has been tested for small scale multi-agent systems: their experiments use 7 agents and up to 50 tasks. Note that we experiment with hundreds of agents and thousands of tasks.

An alternative approach is to use a central mediator. An example of these approaches is *auction* mechanism which has been very popular recently. In auctions, a central auctioneer puts tasks in a market and agents can bid for them according to their cost functions. Auctions come in two flavors: single-task and combinatorial. In *single-task auctions*, agents are uncertain about the preferences of other agents and about future contracts. Hence, agents make short-sighted commitments. *Combinatorial auctions* [14] can theoretically find optimal solutions but they suffer from some major deficiencies: an agent needs to compute a strategy for bidding, and the number of combinations of items (known as bundles) grows exponentially as a function of the size of a bundle, which makes the cost calculation for large instances intractable–both at the bidder's end and at the seller's end (where winners are chosen in the auction clearance).

Reallocation mechanisms are generally used to improve the quality of solutions obtained through centralized approaches. Peer-to-peer task reallocation is equivalent to a local search where every move or exchange between the agents decreases the cost or increases the utility. It has been shown that global optimum can be reached in finite number of steps if the agents could have a complete look ahead [11], where each step is an *O-contract* which moves a single task possibly with some payment. However, as we mentioned earlier, complete look ahead is computationally intractable.

An alternative is to use *OCSM-contracts*: an OCSMcontract is a member of the set formed by the cross-product of a number of different types of contracts: original contracts, cluster contracts, swap contracts and multi-agent contracts. For detailed explanation of these contracts, see [11]. Globally optimal solution can be reached in a finite number of steps using OCSM-contracts and a simple greedy strategy without requiring look ahead [11]. The problem with this approach is two fold: the number of steps required is exponential and evaluating contracts at each step is computationally intractable. There is a need to study strategies for distributed negotiation short of exhaustive enumeration and greedy exchange.

It is interesting to note that Sandholm and his group, who have done considerable research in contract evaluation based on marginal costs, suggest a paradigm shift to centralized or mediated clearing to overcome the inefficient solutions negotiated by myopic agents [12]. However, our work is motivated by the conjecture that complete look ahead may not be necessary for efficient solutions; we conjecture that approximate, partial lookahead will support a decentralized approach that is both scalable and efficient.

3. Approach

MALA works in two-phases (see Figure 1):

Planning Phase: In the planning phase, each agent computes a preferred tour. Such a tour provides the agent a means of doing a look ahead to determine its utility for a potential task.

Negotiation Phase: In the negotiation phase, an agent attempts to trade tasks with other agents in order to sell unwanted tasks and acquire preferred tasks. Each trade is pair-wise and is evaluated based on a marginal cost analysis done by the two agents involved in the trade.

We now define the problem formally and provide the details of the U-TSP based algorithm and the negotiation strategy.

3.1. Problem Definition

Let T be a set consisting of n tasks, and A be a set consisting of m agents such that $n \gg m$. Assume a k-dimensional Euclidean field on which the tasks in T and the agents in A are located. Their distribution across the field is uniform random. Let T^* be the set consisting of finite sequences of tasks. Each sequence corresponds to a possible tour for an agent, and the tasks in a sequence are said to be assigned to the agent. A task allocation is a mapping $A \rightarrow T^*$ such that the intersection between each of the sequences in the range is empty. In other words, each task is uniquely assigned to an agent.

According to the taxonomy presented in [6], the problem we are addressing may be characterized as follows. Tasks are *single-agent*, i.e., tasks require just one single agent to service. Agents are *single-task*, i.e., agents can service only one task at a time. The approach is *time-extended* since the agents plan ahead and may include more than one task in their schedule.

We start with a random allocation R. The goal is to find a task allocation O such that the maximum number of tasks can be serviced by the agents. Note that starting with a random allocation ensures that each task is initially allocated to some agent. Each agent is subject to a fuel constraint which limits the total distance it can travel. It is obvious that some tasks may be impossible to service due to the fuel constraint, even given an optimal allocation.

We assume that the field is static in the sense that neither the agents, nor the tasks, move, appear or disappear; the communication between agents is reliable; and there is no central mediator. These assumptions ensure that the only uncertainties agents face are those related to other agents' preferences. The simplification enables us to focus on the effect of limited look ahead, measured in terms of scalability and quality of solution metrics.

3.2. Mechanism Design

A market-based approach has to fix the rules of the game (mechanism). Mechanism design critically affects potential outcomes: properly designed mechanisms play a significant role in restricting the solution space to a space that mostly contains desirable outcomes [16]. Our market uses the following mechanism:

- Each task has a constant utility associated with it. This provides each agent the incentive to maximize the number of tasks it services.
- Each task which remains un-serviced incurs a penalty (constant p here) for the agent assigned to it. Thus, an agent has the incentive to actively sell-off tasks which it does not expect to be able to service.
- Agents *offer* tasks which they own to the market. Such offers may include side-payments as inducements.
- Agents may *bid* for tasks that are available in the market.
- Agents may *accept* a bid.
- Agents can express their interest in buying a task which is owned by some other agent but is not available in the market. Such offers are called *hostile offers*. Hostile offers are accompanied by the offer of a payment as inducement.

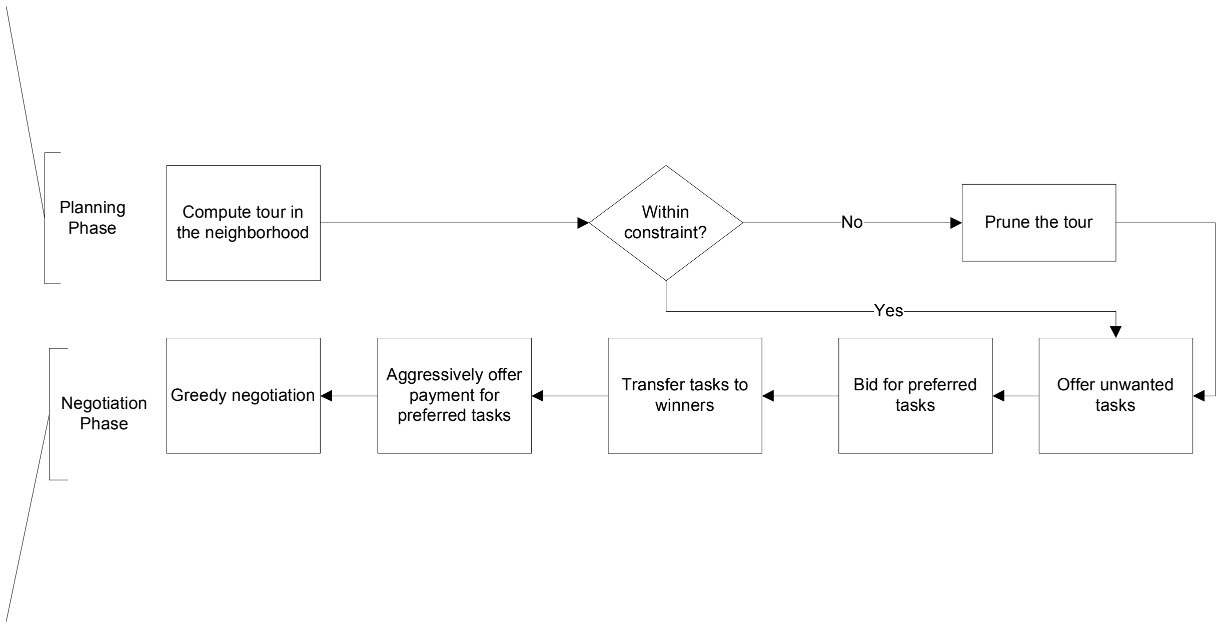


Figure 1. Overview of the approach.

3.3. Modified Universal TSP

In the planning phase, each agent computes a preferred tour in its neighborhood using the Universal TSP (U-TSP) algorithm [9] such that the tour maximizes its reward for the given fuel constraint. We express the fuel constraint in terms of total distance an agent may travel. Since the agent is required to return to its origin, an agent's neighborhood is defined as the region within the radius $d/2$, where d is the agent's fuel constraint. Once a tour for all destinations in the neighborhood has been computed, the U-TSP algorithm enables a linear time computation of the cost of a subtour. This enables a marginal cost analysis in linear time.

Using the U-TSP algorithm provides an $O(\log n)$ approximation for the cost of a the subtour. On the other hand, using a 2-approximation TSP algorithm would produce an approximation that is at most two times from the optimum cost of the tour. However, in this case evaluating each trade requires up to four computations, each of which costs $O(n \log n)$ [2, 4]. This is a significant cost for large problem instances and hinders scalability. Thus we have to make a trade-off between solution quality and scalability.

In order to try to reduce the error in the U-TSP approximation, we modify it as follows. Note that the U-TSP algorithm is based on a tree data structure and involves some nondeterminism during its computation. In order to improve the tours, an agent computes a tighter tour *on the same points* using the 2-approximation algorithm [3]. This computation is used to heuristically guide the U-TSP algorithm's initial computation. With such guidance, we observed significant improvements in the quality of the solutions provided by the U-TSP algorithm.

The need for tight tours is fairly obvious. Our global objective is to maximize the number of tasks carried out given that each agent has a fuel constraint. This is helped by finding low-cost tours for each agent: in general, the global objective is positively correlated with agent allocations which minimize the cost of its tour.

The complete algorithm is described in Algorithm 1.

Algorithm 1 - Modified Universal TSP Algorithm

```
T = Tree structure from U-TSP algorithm
R = 2-approximation tour

makeTour (Node T)
  if (T.hasChildren)
      for all (v in t.children)
          - $w$ = makeTour(v)
          - rank($w$) = $\arg\max_{wi \in w}$ (indexOf($wi$, $R$))
          - list.add($w$)
      sort list in ascending order of rank
      for all (v in list)
          add elements of v to the list Tour
      return Tour
  else
      return list containing T
```

While we constrain an agent's tour to tasks within its radius, the tour obtained by our modified U-TSP may exceed the agent's fuel constraint (and generally will). To prune this tour, we experimented with two approaches. We first tried a greedy algorithm: find a task with the highest marginal cost at each step and remove it from the tour; repeat until the tour cost is just within the agent's fuel constraint.

We then tried a more sophisticated approach which exploits the tree-based data structure of the U-TSP algorithm. The core idea behind U-TSP algorithm is to divide the field into small regions and progressively build a tree of tasks bottom-up by selecting coarser regions at each subsequent level. The intuition behind our algorithm is to exploit the structure provided by this tree to remove the smallest region from the tour such that the resulting tour is just within an agent's fuel constraint.

Specifically, our algorithm starts from the root of the tree and removes branches which correspond to chunks of region from the tour until the tour cost falls below the agent's fuel constraint. At this point, it "zooms down" on that chunk, keeps removing finer chunks until the tour cost falls below d. This repeats until it finds a tour such that the tour's cost is just below the fuel constraint d. From experiments, we found that the second algorithm produces marginally better tours than the first one. The second algorithm is described in Algorithm 2.

Algorithm 2 - Pruning Tours

```
pruneTree (Node T)
  if (T is leaf)
      remove T from tour
  while($cost >$ CONSTRAINT)
      - Find the branch j having the least
          reward with respect to cost
      - Remove tasks in the branch from tour
      - Compute $cost$ of the remaining tour
  Restore the tasks in branch j to the tour
  pruneTree (j)
```

3.4. Negotiation Strategy

Once all agents have computed a preferred tour, they negotiate with each other towards their preferred tour. At step 1, agents put tasks they do not want in the market (recall that an agent will be penalized for not servicing tasks assigned to it in the final allocation). Let the set of tasks that were initially assigned (by random allocation) to an agent be RA, and the set of tasks in its preferred tour be PT. Then the agent wants to get rid of $RA \setminus PT$, and acquire $PT \setminus RA$ from the market.

If only one agent is interested in an offered task, it simply takes the task. If multiple agents are interested in a task, the owner agent starts an auction realizing the potential to gain utility. After such tasks have been moved, we have one the following situations for each task t:

1. Only one agent prefers t and the task is assigned to that agent. In this case, no further action is necessary.

2. No agent prefers to have t but it is assigned to one of them. In this case, the owner computes the maximum side-payment it can offer for t and puts this offer into the market. Each agent for whom the task is in its range uses this side-payment to compute its marginal gain.

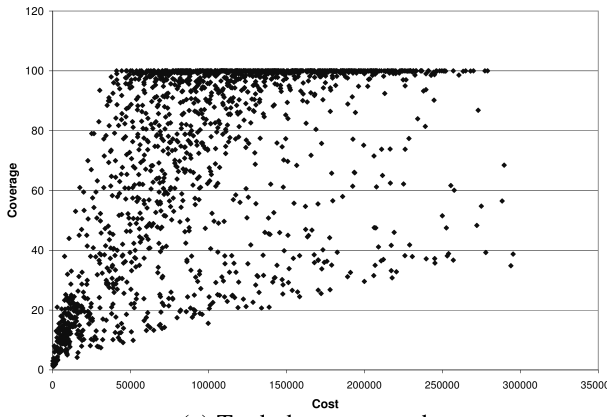
3. Multiple agents prefer t and one of them owns it. Observe that t may not be available in the market if random allocation assigns it to an agent that prefers t. This agent may not have the highest incentive to service t, though. Each of the remaining interested agents puts a hostile bid in the market offering a price it is willing to pay based on its marginal cost analysis. The owner decides whether to accept the offer based on its own marginal cost analysis.

Thus far marginal cost analysis has been based on agents' preferred tours (See Figure 1: we are now at the point just before the box 'Greedy negotiation'). We assume that in the face of uncertainty, agents remain "optimistic" about reaching their preferred tour. Now the agents realize that they may not reach the preferred tour and become "pessimistic." In this case, we assume agents resort to a greedy strategy and analyze contracts myopically, based on their current assignment.

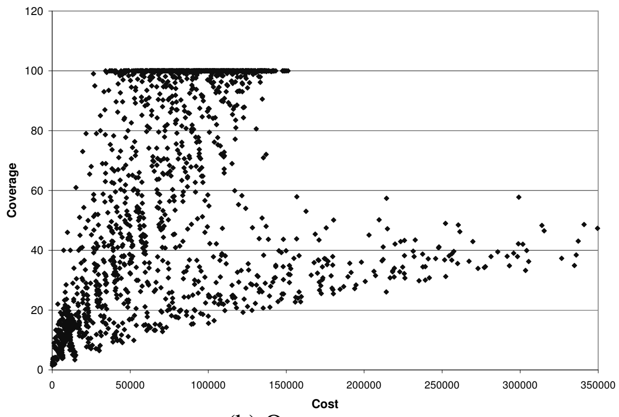
4. Results

We tested the effectiveness of using MALA by experiments done on a custom-designed simulator. The simulator is synchronous: at each time step, every agent is allowed to execute some action. We generated 1730 problem instances, each representing a different combination of number of agents (from 20 to 200 in increments of 20), number of tasks (from 100 to 2000 in increments of 100) and fuel constraint (from 100 to 1900 in increments of 200). We also imposed the requirement that the number of tasks must be at least three times the number of agents in every instance.

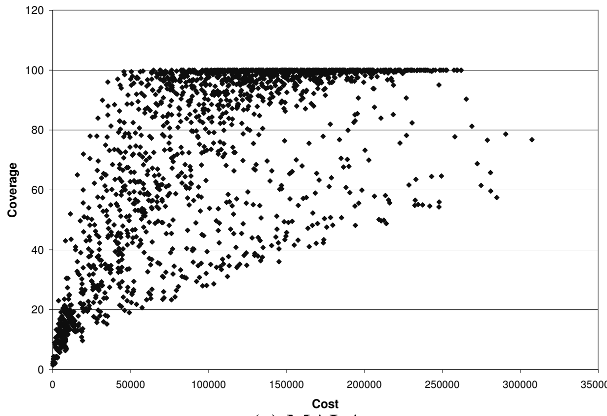
We ran the simulations on the same test instances for each of three approaches: MALA, centralized O-contracts, and Traderbots. We measured the (simulated) computation time, the number of messages required, the number of broadcasts required, coverage and cost of the tours. Our definition of coverage is as follows. Let d be the fuel constraint for an agent. Then every task located outside the radius $d/2$ of that agent is trivially infeasible (for the agent).



(a) Traderbots approach.



(b) O-contracts.



(c) MALA.

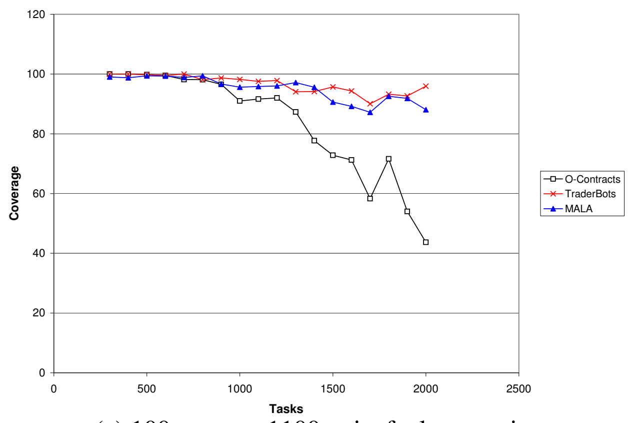
Figure 2. Solution quality as coverage vs cost for all instances.

Coverage is the number of not trivially infeasible tasks that have been allocated to some agent as a fraction of the total number of tasks. Note that servicing a task may be infeasible but not trivially unfeasible. In MALA, this may happen in two ways: firstly, because the initial random allocation assigns an infeasible task to an agent which the agent is not able to subsequently get rid of; or secondly, because of the use of the greedy strategy by a pessimistic agent. This means that our coverage metric introduces a possible bias against MALA: unlike MALA, neither of the other approaches prunes tasks within the fuel range of agents to arrive at their tours. Thus the tours proposed by the other approaches may be more likely to be infeasible, which would make our estimate of their coverage relatively generous. We have not studied the results for this bias.

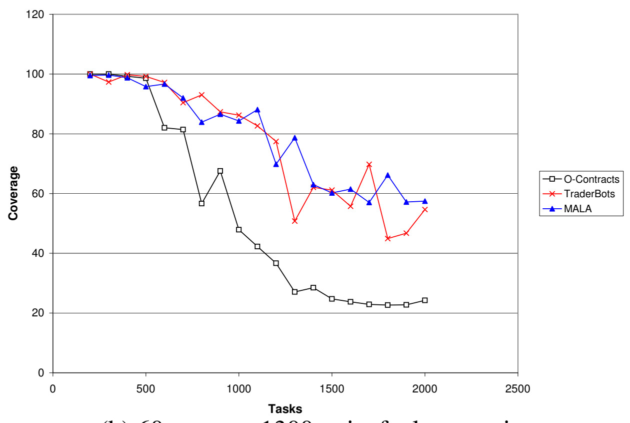
Figure 2 (a,b,c) shows scatter plots of coverage against cost representing all problem instances. Essentially, this plot represents the quality of solution. High-quality solutions provide high coverage for lowest possible cost (the upper-left portion of the graph) and this is where we would like more points to appear.

The O-contracts approach results in a significant number of points in the high cost, low coverage (lower-right) section of the graph. The greedy, multi-agent approach does better than the O-contracts approach but MALA fares significantly better than both–as evident by the density of points in the upper-left section of the graph. The lower periphery of the plots warrants careful observation. In plot (c), the lower periphery has a higher slope compared to those in plots (a) and (b) and thus, seems to evade the lower-right region. For example, at 100000 units of cost, the lowest point in (c) is at around 28% coverage while corresponding numbers for (a) and (b) are 15% and 16%. Similarly, at 150000 units of cost, the lowest point in (c) is at around 41% coverage while corresponding numbers in (a) and (b) are 22% and 23%. This observation supports our hypothesis that the approximate look ahead helps in avoiding the worst solutions.

Figure 3 (a) shows a plot of coverage versus number of tasks, for a specific combination of number of agents and fuel constraint (100 and 1100 respectively) while (b) shows the same for 60 agents and fuel constraint of 1300. Intuitively, coverage will decrease with increasing number of

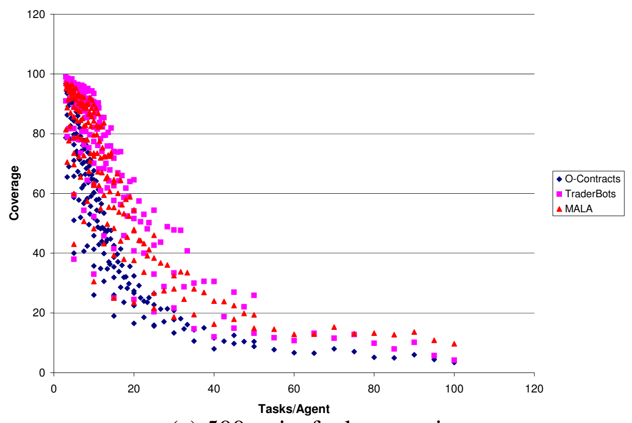


(a) 100 agents - 1100 units fuel constraint

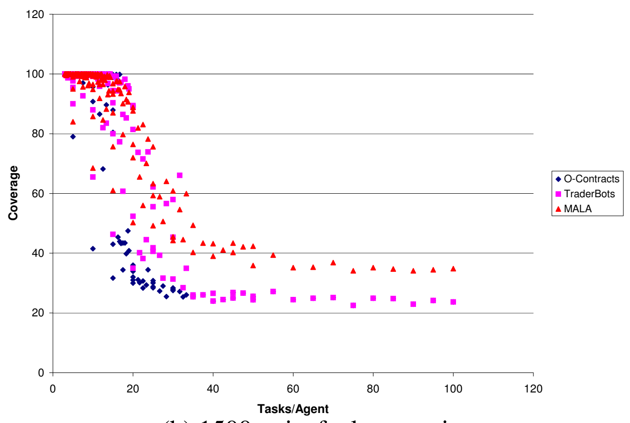


(b) 60 agents - 1300 units fuel constraint

Figure 3. Behavior of coverage with respect to tasks.



(a) 500 units fuel constraint



(b) 1500 units fuel constraint

Figure 4. Behavior of coverage with respect to the ratio tasks/agents.

tasks keeping other factors constant, and this plot indicates how different approaches fare relative to each other. Coverage drops rapidly for O-contracts based approach as number of tasks increase. This suggests that O-contracts based approach is more likely to get stuck in local optimum. The curve for MALA shows a much lower slope. Interestingly, the other distributed approach has a very similar curve to that of MALA.

We analyzed coverage as a function of the ratio of tasks per agent, keeping the agents' fuel constraint constant. The difficulty of satisfying a problem instance increases as a function of this ratio (given constant fuel), and therefore the coverage drops. Figure 4 shows a scatter plot for all three approaches for two different fuel constraints. It is quite clear that both the distributed approaches provide better solutions than the central approach. For the most difficult instances–those with a density ratio of 50 or more–MALA dominates the other two approaches in both the plots. This again suggests that MALA avoids the worst solutions.

Figure 5 shows the average coverage as a function of the fuel constraint (the average is taken across all combinations of numbers of agents and tasks). The best fit model for these curves appear to be a power law (R-squared for the best fits: 0.45 for O-contracts, 0.44 for TraderBots, and 0.60 for MALA).

To analyze these results statistically, we performed a number of paired t-tests (recall that we ran simulations for all approaches for the *same* problem instances). For coverage as a function of fuel, the TraderBots approach does better than MALA for fuel constraints below 900 ($p < .001$), while MALA outperforms TraderBots for fuel constraints greater than 1500 ($p < .001$). The two approaches are comparable between fuel constraint of 900 and 1500. Observe that MALA provides good solutions in comparison to the other two approaches despite them employing a 2-approximation TSP algorithm for evaluating contracts.

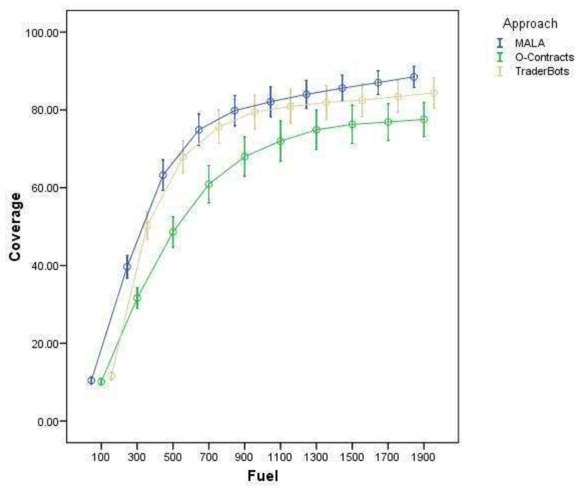


Figure 5. Coverage vs Fuel across all combinations of number of agents and number of tasks. 95% Confidence Intervals around the mean are shown.

Figure 6 compares the communication cost of the two distributed approaches, MALA and TraderBots, in terms of messages (point-to-point) and broadcasts (offers made in the market). Clearly, MALA requires fewer messages and broadcasts to reach its solution: a paired t-test shows that the difference is significant ($t = 42, p < .001$ for messages, and $t = 37, p < .001$ for broadcasts). These results can be explained by observing that in the planning phase of MALA, agents are selective in determining their preferred trades, and therefore require less communication. The result supports the importance of look ahead. Because the O-contracts approach is a centralized approach, obviously there is no communication overhead during the computation. However, a burst of communication is expected in the beginning to gather preferences from agents, and after the computation in order to propagate the allocation to agents. Gathering preferences requires scheduling messages of all the agents; this could be quite expensive and involve a lot of messages. Moreover, communication with the centralized server are also likely to be expensive as the server could be a hotspot.

We did all our simulations on a single machine. In our approach, we observed that significant computation time is spent in computing U-TSP based tours. In a real distributed environment, this time would be spread across all agents executing in a parallel fashion. On the other hand, the O-contracts based approach is completely centralized and has a very high computation time. The TraderBots approach does not involve computing tours and does better in terms of computation time per agent. In real settings, though, the TraderBots approach would suffer from high convergence time due to very high volume of messages and broadcasts required. We believe there is ample evidence from the results that the investment in computing a preferred tour ahead pays off significantly both in terms of solution quality and the number of messages exchanged.

5. Conclusion and Future Work

We proposed a novel market-based approach to solve multi-agent task allocation, whereby an agent has a limited look ahead. Agents trade tasks based on their estimates of the marginal cost or payoff of a trade. We proposed using a modified U-TSP algorithm to enable agents to achieve such look ahead because U-TSP based marginal cost computations are linear in time and hence allow the agents to process large number of messages at each step. This enables our algorithm to be scalable to large instances of the problem. The results of our simulations support the conjecture that such limited look ahead may provide an effective and scalable solution to the problem.

By using an initial allocation, we defined the problem as one of task reallocation. However, the choice of random allocation may not be the best. An alternative initial allocation could be obtained by using an auction mechanism. It is possible that such an allocation could further improve the solution quality while lowering computation and communication cost. Furthermore, an allocation obtained through auctions allows us to relax the assumption of 'known' world, and reliable communication in the whole field. Each agent will bid for and trade tasks, and communicate with other agents located only in its neighborhood.

To evaluate the pay-off from look ahead, we assumed a two dimensional world and reliable communication. We believe this work can be readily extended to a k-dimensional Euclidean world. However, in certain applications, such as search and rescue operations in an urban terrain, Euclidean metric may be inapplicable. In some cases, a City Block or *Manhattan metric* may be more appropriate. In this case, using the Euclidean assumption in MALA may still provide a reasonable approximation that may be adjusted: the Manhattan distance between two points is at most $\sqrt{2}$ times the Euclidean distance between them, and on average, as a simple calculation shows, it is 1.28 times the Euclidean distance. However, note that obstacles could introduce a complex topology that cannot be captured by the Manhattan metric; in this case, our results may not hold.

It is important to reiterate that MALA does not lead to optimal solutions as the look ahead itself is approximate. Even if the look ahead were optimal, this preference would be unknown to other agents. Moreover, multiple agents may have conflicting preferences, making global optimization difficult. In our approach, look ahead provides temporary "optimism" to the agent (in terms of allocation they

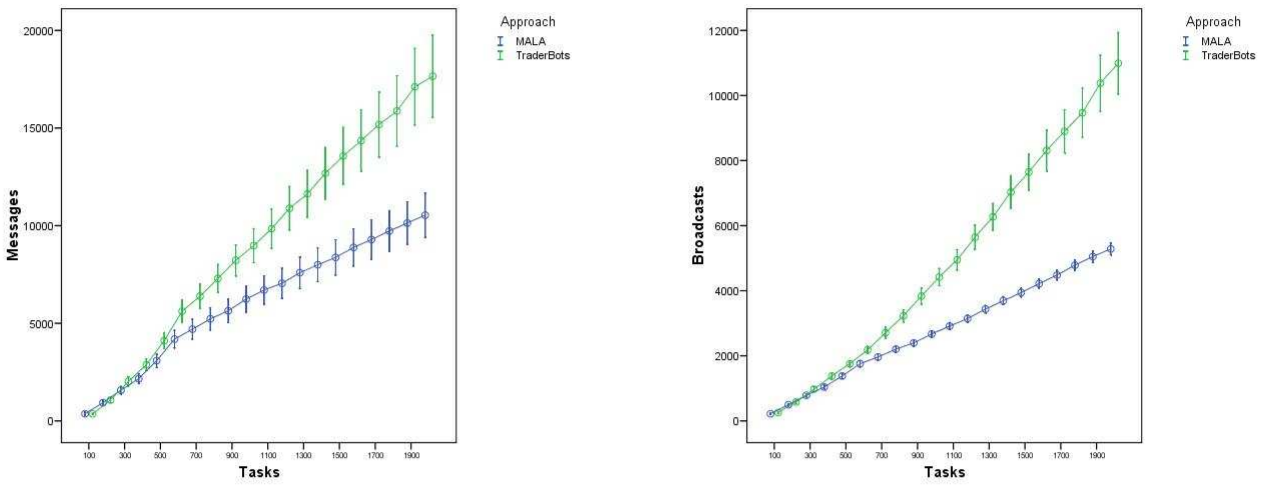


Figure 6. Number of messages and broadcasts required vs the number of tasks across all combinations of number of agents and fuel constraint. 95% Confidence Intervals around the mean are shown.

have reached a milestone or a higher level); once they realize that they may not have what they prefer, agents switch to "pessimism" as a fall-back strategy. In our implementation, pessimistic agents adopt a greedy approach. We found some literature in economics on modeling optimism or pessimism due to uncertainty in game-theoretic settings [13]. We believe there is potential future work in this direction.

We plan to extend our approach to handle dynamic environments where tasks are allowed to appear and disappear, and agents may have limited mobility. Also, there are quite a few applications which impose constraints or dependencies between the tasks. The challenge in this case is to represent and evaluate bids for interdependent tasks. Another generalization of the current problem is as follows: instead of each task requiring a single agent, a task may require multiple agents to service it. Finally, we plan to test the behavior of MALA with imperfect communication and heterogenous agents (such as agents having different fuel constraint) which can introduce further uncertainties into the problem.

Acknowledgements

We would like to thank Chandra Chekuri for suggesting the use of the Universal TSP algorithm and for providing useful insights into alternative approaches for planning tours. We would also like to thank Vijay Anand Reddy for suggestions about the relation between Manhattan and Euclidean metrics.

References

[1] A. Ahmed, A. Patel, T. Brown, M. Ham, M.-W. Jang, and G. Agha. Task assignment for a physical agent team via a dynamic forward/reverse auction mechanism. In *The International Conference of Integration of Knowledge Intensive Multi-Agent Systems KIMAS '05: Modeling, Evolutions and Engineering*, pages 311–317, April 2005.

[2] M. Andersson and T. Sandholm. Time-quality tradeoffs in reallocative negotiation with combinatorial contract types. In *AAAI/IAAI*, pages 3–10, 1999.

[3] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms, 2nd edition.* MIT Press, McGraw-Hill Book Company, 2000.

[4] M. B. Dias, B. Ghanem, and A. T. Stentz. Improving cost estimation in market-based coordination of a distributed sensing task. In *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3972 – 3977, August 2005.

[5] M. B. Dias and A. T. Stentz. Opportunistic optimization for market-based multirobot control. In *Proceedings of the 2002 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '02)*, volume 3, pages 2714 – 2720, September 2002.

[6] M. B. Dias, R. M. Zlot, N. Kalra, and A. T. Stentz. Market-based multirobot coordination: A survey and analysis. Technical Report CMU-RI-TR-05-13, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, April 2005.

[7] M. B. Dias, R. M. Zlot, M. B. Zinck, J. P. Gonzalez, and A. T. Stentz. A versatile implementation of the traderbots approach for multirobot coordination. In *Proceedings*

of the International Conference on Intelligent Autonomous Systems (IAS), March 2004.

[8] B. P. Gerkey and M. J. Matarić. Sold!: Auction methods for multi-robot coordination. *IEEE Transactions on Robotics and Automation*, 18(5):758–768, Oct. 2002.

[9] L. Jia, G. Lin, G. Noubir, R. Rajaraman, and R. Sundaram. Universal approximations for TSP, Steiner tree, and set cover. In *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, pages 386–395, 2005.

[10] L. Navarro, R. Grabowski, C. Paredis, and P. Khosla. Millibots. *IEEE Robotics and Automation Magazine*, pages 31 – 40, December 2002.

[11] T. Sandholm. Contract types for satisficing task allocation: I theoretical results. In *AAAI Spring Symposium: Satisficing Models*, 1998.

[12] T. Sandholm. Making markets and democracy work: A story of incentives and computing. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1649–1671, 2003.

[13] B. C. Schipper. The evolutionary stability of optimism, pessimism and complete ignorance. Discussion Papers 68, SFB/TR 15 Governance and the Efficiency of Economic Systems, Free University of Berlin, Humboldt University of Berlin, University of Bonn, University, Nov. 2005. available at http://ideas.repec.org/p/trf/wpaper/68.html.

[14] T. Smith, T. Sandholm, and R. Simmons. Constructing and clearing combinatorial exchanges using preference elicitation. In *Proceedings of the AAAI workshop on Preferences in AI and CP: Symbolic Approaches*, 2002.

[15] P. B. Sujit, A. Sinha, and D. Ghose. Multiple uav task allocation using negotiation. In *AAMAS '06: Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pages 471–478, New York, NY, USA, 2006. ACM Press.

[16] H. Varian. Economic mechanism design for computerized agents. In *Proceedings of the First USENIX Workshop on Electronic Commerce*, July 1995.

[17] R. Zlot, A. Stentz, M. Dias, and S. Thayer. Multi-robot exploration controlled by a market economy. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2002.