AN INTEGRATED APPROACH TO SOLVING THE REAL-WORLD MULTIPLE TRAVELING ROBOT PROBLEM

Sanem Sariel[*] Nadia Erdogan[*] Tucker Balch[+]
e-mail: sariel@itu.edu.tr e-mail: nerdogan@itu.edu.tr e-mail: tucker.balch@gatech.edu
[*] *Istanbul Technical University, Electrical and Electronics Faculty, Dept. of Computer Eng., 34469, Istanbul, TURKEY*
[+]*Georgia Institute of Technology, College of Computing, Atlanta, GA, USA, 30332*

Key words: Multi-robot systems, real-world multiple traveling robot problem, incremental task selection, robustness

ABSTRACT

In this study, we analyze the real-world Multiple Traveling Robot Problem (MTRP) and propose an integrated approach to solve this problem in real time. The MTRP is a generalization of the well-known Multiple Traveling Salesman Problem and is solved by a multi-robot team. In the MTRP definition, target locations should be visited by the robots in the team while optimizing an objective function. Since the real world is beyond the control of robots, in most cases operations research (OR) solutions are not directly applicable due to either robot hardware/software limitations or environmental dynamics. In this paper, we analyze the MTRP from the real-world perspectives. In our solution, dynamic task selection, distributed task allocation and contingency handling mechanisms along with the low-level robot controllers and the motor and sensory modules are integrated into each other to solve the real-world MTRP. Target allocation and route construction is integrated into each other by an incremental assignment approach. Real-time situations and contingencies that change the problem instance are handled at the same time. Empirical evaluations of the system performed on the WEBOTS robot simulator reveal the efficiency of the integrated components of our approach.

I. INTRODUCTION

The single robot exploration problem, a variation of the Traveling Salesman Problem (TSP), is to find the minimum cost traversal of a given number of targets without considering the return cost from the last target to the initial location for a single robot [4]. The problem can also be stated as finding the minimal Hamiltonian path on a given fully connected graph with all nodes to be visited. The Multiple Traveling Robot Problem (MTRP) or the multi-robot multi-target exploration problem is a more general version of the TSP in which there is a team of robots to visit targets at least once (ideally at most once). In the MTRP, besides the quality of the constructed paths of the robots, allocation of the targets is quite affective on the overall solution quality. Different types of complementary objectives to the main goal may be selected to optimize the performance for this problem as in scheduling problems. These optimization objectives may be set for the total path length, the time, the average energy consumed, makespan, etc. For example, in search and rescue operations, the mission is highly challenging with hard-time constraints and the success is hugely crucial. In space explorations, the mission is exploring the unknown outer space to collect scientific data. In this domain, robots communicate through the satellite links that are highly prone to communication failures and latency. Instead of optimizing the time, the battery/fuel life of the robots may be optimized in this domain. Based on the selected objective function, the cost evaluation may need to be designed differently.

The optimal results for the MTRP can be obtained using Integer Programming (IP) formulations. However, these approaches may become impractical when the size of the mission is even moderate or the cost values change frequently due to the uncertain knowledge, changes in the environment (including failures) or the changing structure of the mission (e.g. online tasks). Robots also have continuous path planning burdens for the target sets in dynamic environments. Expensive computational efforts made for allocating the targets may become redundant.

We are in favor of the distributed implementations, where target allocations are made by the robots themselves due to the real-world limitations. We propose an incremental task allocation and route construction approach and a contingency handling mechanism integrated into each other on the task allocation layer. Those components are also integrated into the separate low-level robot localization, mapping and multi-target path planning layers. Since the schedules are subject to change in the real world during runtime, the incremental allocation approach ensures task allocations are made in an efficient and scalable way reducing redundant efforts. The contingency handling mechanism integrated into the target allocation mechanism provides efficiency in detecting and recovering from failures and unplanned events.

The MTRP is analyzed in different multi-robot systems by using both combinatorial [1][2] and single-item auction

methods in literature. Since we are interested in an efficient implementation and try to reduce the computational and communication efforts, we focus on the single-item target allocation approaches. The Prim Allocation method [3], one of the easy to implement and efficient single-item target allocation methods generates an MSF of targets and robots. Each robot offers an auction for a target and one of the targets are allocated at each round in the implementation. Whenever the world knowledge changes, the remaining unvisited targets are reallocated using the same algorithm. Like Prim's Algorithm, the Prim Allocation method is bounded by 2*OPT for the MTRP. The Prim Allocation approach offers ways to allocate targets. However, our focus is on integrating both efficient target allocation methods and real robot implementations, and furthermore dealing with the real-world ingredients of the MTRP. We compare our approach with Prim Allocation in our earlier work [7],[9]. In this study, we analyze the performance of the integrated components of our proposed approach.

II. PROPOSED APPROACH

In practical applications, computing the true optimal solutions is not required due to several reasons [6]. Those reasons may be the incorrect modelling of the underlying problem (targets) and the lack of sufficient real-time to find the optimal solution. These are common cases in robot applications besides the failures and the other real-world contingencies. To meet all these limitations, we propose a dynamic and distributed task allocation scheme, to coordinate robots that cooperate to fulfil different parts of a mission. Dynamism is achieved through the incremental selection and allocation of the targets. Distribution is achieved by allowing each robot to select its own candidate tasks and by further selecting appropriate robots for the tasks to be executed. Since the selection is incremental, the target allocation is interleaved with the route construction.

The proposed approach can efficiently respond to real-time events and the solution quality is maintained simultaneously with the real-time task execution by the cooperative processing of the integrated components of the system. We propose a general mechanism for multi robot cooperation for the MTRP but not necessarily specific heuristic functions to solve the problem although their success is validated previously [9].

DYNAMIC TASK SELECTION AND INCREMENTAL ALLOCATION

Since there is a tight connection between route generation and allocations, robots initially generate rough routes (schedules) in our heuristic approach. Next, each robot (r_j) selects its most suitable target among the targets in the rough target set (T_{Rj}). T_{Rj} is constructed by selecting the targets close to robot r_j, among unvisited targets (T_U) according to Eq. (1), where *dist* function returns the Euclidean distance between two points. Targets in T_{Rj} are considered as the candidate targets for robot r_j. Therefore, before selecting the most suitable target, each robot constructs these rough route sets. This heuristic does not compel an actual commitment, and the targets in the rough routes are not necessarily assigned to the corresponding robots in the future auctions. Instead, it provides a global view to the problem from a local perspective.

$$reldist(r_j, t_i) = dist(r_j, t_i) - \min(dist(r_k, t_i)), \{\forall k \neq j\} \quad (1)$$
$$T_{Rj} = \cup t_i, reldist(r_j, t_i) < 0, \forall t_i \in T_U$$

We use the simplest cost function to evaluate the targets in T_{Rj} for each robot. This cost function for robot r_j and target t_i is simply evaluated as in Eq. (2) by considering the distance between the target and the robot.

$$c_{ji} = dist(r_j, t_i), t_i \in T_{Rj} \quad (2)$$

Each robot executes Algorithm 1 to generate its rough schedule, and then selects the most suitable candidate task (t_S, the most suitable target among the rough schedule targets) to perform.

Algorithm-1. Rough Schedule Generation, r_j

```
name: MTRP-FormRoughSchedule
input: T_U , output: T_Rj and t_S
T_Rj = φ  (a heap with the key as the task cost)
t_S = φ
while ( T_U ≠ φ )
    if t_i is in the rough schedule region
        evaluate c_ji and insert t_i into T_Rj
if || T_Rj ||> 0
    t_S = top(T_Rj)
```

Algorithm-2. Incremental Task Selection, r_j

```
name: MTRP-DPTSS
input: T_U , output: action to be performed
[ T_Rj , t_S ] = MTRP-FormRoughSchedule( T_U )
if t_S ≠ φ
    if t_S is the current task
        continue with the current execution
    else
        if t_S is an available task
            offer an auction
        else switch executing the awarded task
else stay idle
```

Algorithm 2 forms the main loop of the incremental task allocation and it is called in the beginning of the mission execution and whenever the world knowledge of the robot changes. Each robot executes the same algorithm concurrently until the end of the mission, when all traversable targets are visited. The given algorithm may be used to allocate all targets from scratch. However, an incremental approach eliminates the redundant allocations for dynamic environments. The cost function design can be determined based on the capabilities of the robot. The cost function that we use can be successfully implemented for very small robots, as in our experiments.

DISTRIBUTED TARGET ALLOCATION

After selecting the most suitable target for itself, each robot announces its intentions by a single-item auction. Selection of the best robot for a task is performed by using the Contract Net Protocol (CNP) in our approach. Although CNP presents the formalism on the relationships between managers and contractors, some simple decisions are left to the designer. Most auction-based task allocation schemes offer solutions for allocating one/a subset of task(s) of the overall mission. However, there is usually little information about when task announcements and reassignments are made.

Algoritma-3. Main Asynchronous Procedures for some events, r_j

```
if auction negotiation deadline is reached
     end auction
     award the best bidder or begin execution
if an auction message for t_k from r_l is received
     Perform the corresponding MTRP-Response (Path/Time)
if an award message is received and || T_Rj || = 0
     begin executing the task
if the world knowledge is changed affecting T_Rj
     cancel auctions or executions
```

Algoritma-3'. Response from r_j for the auction of t_k to the auctioneer r_l - Path Minimization

```
algorithm: MTRP-Response-Path
if  auction/execution is in progress and c_ji > c_lk
     cancel auction/execution
if c_ji < c_lk
     send bid value for t_k
```

Algoritma-3". Response from r_j for the auction of t_k to the auctioneer r_l - Time Minimization

```
algorithm: MTRP-Response-Time
if  auction/execution is in progress and
c_ji > c_jk || c_ji > c_lk and t_k or r_l is close to T_Rj
     cancel auction/execution
if c_ji < c_lk
     send bid value for t_k
```

The approach we propose allows multiple auctioneers and winners for different tasks, depending on the optimization objective. Algorithm 3 presents the responsive behaviours for asynchronous events as distributed target allocation procedures. In the case of the total path length minimization objective, when there are relations between targets ending one auction at a time results in better performance. When the robots receive task execution intention messages as auctions, they either send their cost values as bids for the announced targets or send warning messages to the sender robots by the system consistency procedures as explained in the following section.

III. PLAN B PRECAUTIONS

The contingency handling mechanism in the proposed approach is used to detect failures or contingencies and to recover from them. Contingency handling is provided as an integrated component to the target allocation mechanism to ensure robustness. It is formed by two components: *The Model Update Module* and *The System Consistency Checking Module*. *The Model Update Module* uses incoming information and the own perception data to update the world model of the robot. *The System Consistency Checking Module* provides warning messages to keep the system consistent. Related to the contingent situations, appropriate consistency checking and update procedures are activated to either correct the models, or initiate recoveries.

REPRESENTATION OF THE SYSTEM MODEL IN EACH ROBOT'S WORLD KNOWLEDGE

Each robot keeps the models of the system tasks and other robots in its knowledge base. Models of different robots may become inconsistent because of uncertainties, incomplete knowledge, assumptions, etc. It is not always possible to share common world knowledge in decentralized systems as in the case of ours. Task models are stored as Finite State Machines (FSM) where each task can be in a different state. The FSM for the task states are illustrated in Figure 1. The state transitions are activated by *The Model Update Module* for FSMs. Different task states that tasks can be represented as follows:

- *available*: This is the initial state of tasks that are neither in execution nor in auction.
- *invalid* (interpreted as state *available*): This state is activated whenever there is incomplete information regarding a task which is previously being *auctioned* or *executed*.
- *self_auctioned*: A task in this state is under auction negotiation process managed by the corresponding robot as an auctioneer.
- *others_auctioned*: A task in this state is under auction negotiation process managed by another robot as an auctioneer.
- *awarded*: A task is being represented with this state, if the robot either selects itself or being

awarded at the end of an auction negotiation process.

- *self_inexec*: A task in this state is being executed by the corresponding robot in a coalition and synchronized or the coalition involves only the robot itself.
- *others_inexec*: A task in this state is being executed by other robots.
- *achieved*: This is the final state of the tasks that are achieved.
- *unachievable*: This state is used for the tasks that are not traversable or achievable.

Robot models are also stored in FSMs where each robot model has a state which is assumed by the corresponding robot. Robot states are:

- *idle*: A robot is assumed to be in this state when it is believed that it does not execute any task.
- *executing*: A robot is assumed to be running properly and executing a task whenever there is recent evidence.
- *failed*: A robot is assumed to be failed when there is no evidence that it has been running properly for a long time.
- *auctioneer*: A robot is assumed to be auctioned for a task when there is recent evidence.

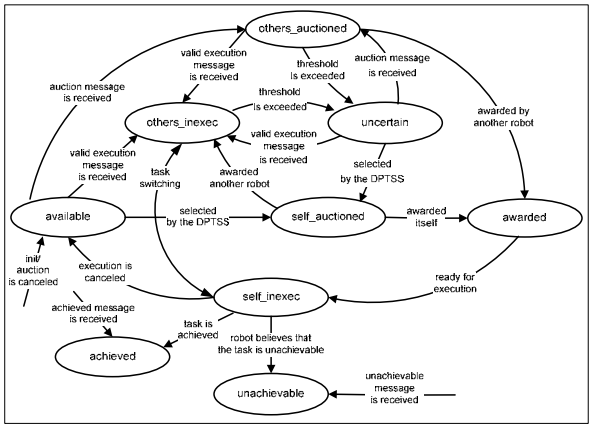

Figure 1. States of the FSM for the task models

MODEL UPDATES AND DUTIES FOR KEEPING THE SYSTEM CONSISTENCY

Recovery operations may include warning other robots about the problem or changing the model accordingly. Inconsistencies usually arise when robots are not informed about the tasks that are achieved, under execution or auction in real-world operations. To keep the system consistency, robots use explicit communication and broadcast the following information:

- new discovered online tasks which are unachieved yet
- task execution messages in predefined time periods (These messages contain the updated cost value and the estimated task achievement deadline information. Therefore, they serve as clues, meaning that the executer robot is still alive and the task is under execution.)
- task achievement message, when the task is achieved
- cancellation message, if the task execution is cancelled
- task invalidation message, when an invalidity is detected

Most of the contingencies are detected by checking the models, and the corresponding model updates are implemented. One standard way of detection of robot failures is sending heart-beat signals. However, in our approach, incoming messages from other robots are taken as clues for being marked as running properly. More complicated prediction models may be used for more accurate failure prediction. Some misleading beliefs such as setting the state of a robot as *failed* although it is running properly may cause parallel executions. This is a desired feature from the mission completion point of view. Designed precautions resolve these kinds of inconsistencies if communication resources permit in later steps.

Note that, information is not assumed to be complete in our approach; robots allocate and execute tasks in a distributed manner. The model updates are implemented in each robot's world model in a completely distributed manner. If communication is available, the system can take advantage of it to ensure the system consistency. Current implementation use explicit communication to detect conflicts and contingencies. However, failures in communication can also be handled by the precaution routines.

IV. EXPERIMENTAL RESULTS AND DISCUSSION

Our real-time dynamic simulation experiments are conducted on the professional mobile robot simulation software, Webots [5]. It contains a rapid prototyping tool to create 3D virtual worlds with robots and objects having physics properties [10]. In our simulation experiments, each environment is represented as a 5m by 5m 3D virtual world where 70mm-size simulated Khepera II robots and objects are located. The environments are randomly generated VRML files containing the robots, the objects and the MTRP targets. Communication is achieved through the wireless links. Emitters in the simulator are configured as having baud rate 9600 and the buffer size 1024B as in the receiver modules.

Robot kinematics calculations are done on the low-level design of the robots by using the odometry information coming from the encoders for both simulation and real-world experiments. Robots perform mapping of the environment by using an occupancy grid approach simultaneously with online localization. In the simulator, slipping and encoder errors are simulated whereas the real

world has its own uncertainties. Due to the slipping errors, as expected from differential wheel robots, there are usually odometry errors.

Different modules on the task allocation layer are integrated with the low-level Sensory Interface, Motor Interface, Motion Model and Mapping modules in the multi-threaded structure.

In the first set of experiments, the scalability of the system for different number of robots is evaluated. Since the proposed approach in this paper is for an efficient multi-robot team, we expect to see a linear decrease in the total path length traversed by the robots with the increasing number of robots as in Figure 2. These results validate our expectations. The overall performance of the total path length traversed by the robot is evaluated in [7,9] with promising results.

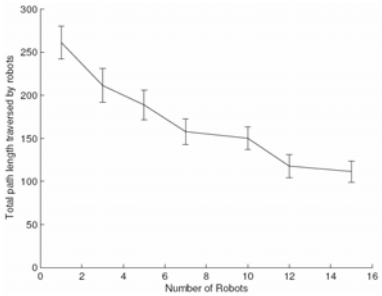

Figure 2. Scalability Analysis for different number of robots

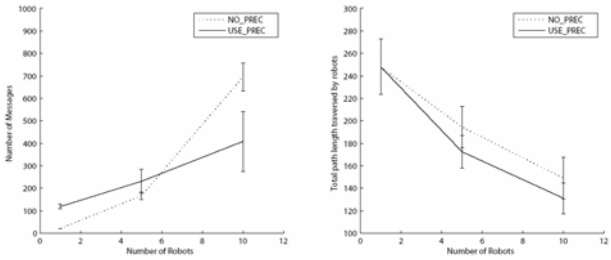

Figure 3. Contingency Handling Mechanism performance results

In the second set of experiments, the contingency handling mechanism performance is evaluated (Figure 3). NO_PREC method does not use the contingency handling mechanism capabilities, while USE_PREC method uses. The figure on the left illustrates the total number of messages sent by the robots. The surprising result here is the increasing number of messages for the NO_PREC case. Although our contingency handling mechanism seems to inject additional messages in the system at first glance (for ensuring the system consistency), these results reveal that it also eliminates the redundancy in the number of messages for multiple bids and auctions. The figure on the right shows the total path length traversed by the robots. The path length is presented by quantized values (each unit is 70mm). The experiments reveal that using the contingency handling mechanisms reduces both the number of messages sent and the total path length traversed by the robots. This result validates the efficiency of integrating the contingency handling mechanism into the system.

V. CONCLUSION

Our solution to the real-world MTRP is integrating the distributed target allocation, the simultaneous route construction, the real-time contingency handling and the low-level robot procedures to make the robots contribute to the overall team objective by visiting the target locations. The target allocation decisions are made by the robots themselves in a distributed manner. Target allocation and route construction is integrated into each other by an incremental assignment approach. Furthermore, real-time situations and contingencies that change the problem instance are handled at the same time. Empirical evaluations of the system are performed on the professional Webots simulator. The presented experiments reveal the success of the approach as a whole with its integrated components and its applicability on even very simple and small robots like Khepera II.

REFERENCES

1. M. Berhault, H. Huang, P. Keskinocak, , W. Elmaghraby, P. Griffin, A. Kleywegt, Robot exploration with combinatorial auctions, IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems, 2003.
2. M.B. Dias, A. Stentz, Opportunistic optimization for market-based multirobot control. *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems,* 2002.
3. M. G. Lagoudakis, P. Keskinocak, A. Kleywegt, S. Koenig, Simple auctions with performance guarantees for multi-robot task allocation, *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems,* 2004.
4. E. L. Lawler, J. K. Lenstra, R. Kan, D. B. Shmoys, *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*, John Wiley & Sons, New York, NY, 1985.
5. O. Michel, Webots: Symbiosis between virtual and real mobile robots. *Proceedings of the First Intl. Conference on Virtual Worlds,* 1998.
6. G. Reinelt, *The Traveling Salesman: Computational Solutions for TSP Applications.* Springer-Verlag, Heidelberg, 1994.
7. S. Sariel, T. Balch, Real Time Auction Based Allocation of Tasks for Multi-Robot Exploration Problem in Dynamic Environments. *Integ. Planning into Scheduling AAAI Workshop,* 2005
8. S. Sariel, T. Balch, A Distributed Multi-Robot Cooperation Framework for Real Time Task Achievement. *DARS8.* Springer-Verlag, 2006.
9. S. Sariel, T. Balch, Efficient Bids on Task Allocation for Multi-Robot Exploration. *The 19th International FLAIRS Conference,* 2006.
10. Webots User Guide 5.1.11., 2006.