Heterogeneous multi-robot mission planning for coordinated tasks execution

Dwiyasa, Felis; Lim, Meng-Hiot; Kang, Pyo; Foo, Ren-Xiang; Teo, Jason Shi-Wei

2020

Dwiyasa, F., Lim, M.-H., Kang, P., Foo, R.-X., & Teo, J. S.-W. (2020). Heterogeneous multi-robot mission planning for coordinated tasks execution. *Proceedings of Soft Computing for Problem Solving 2019*, 1139, 167-173. doi:10.1007/978-981-15-3287-0_13

<https://hdl.handle.net/10356/142110>

https://doi.org/10.1007/978-981-15-3287-0_13

© 2020 Springer Nature Singapore Pte Ltd. All rights reserved. This paper was published in *Proceedings of Soft Computing for Problem Solving 2019* and is made available with permission of Springer Nature Singapore Pte Ltd.

Downloaded on 26 Apr 2023 22:21:56 SGT

Heterogeneous Multi-Robot Mission Planning for Coordinated Tasks Execution

Felis Dwiyasa, Meng-Hiot Lim, Pyo Kang, Ren-Xiang Foo, and Shi-Wei Jason Teo

Nanyang Technological University, Singapore

Abstract The use of robotics and autonomous systems to perform various missions has become increasingly popular and widely accepted. The scope of missions that can be performed by such systems includes complex tasks that require coordination and cooperation between multiple robots, not just simple repetitive tasks. Mission planner plays a vital role in making sure that the tasks performed by the robots are conducted in an efficient manner and meet all the mission requirements. In designing a mission planner, it is crucial to consider not only the functional aspect of the system, but also its modularity, scalability, and interoperability. In this paper, we proposed an architecture of a heterogeneous multi-robot mission planner system that is based on Robotic Operating System (ROS). The system is designed to support heterogeneous multi-robot, where there are several types of robots having unique sets of capabilities and properties, which contributes greatly to the complexity of the mission planner.

1 Introduction

Various sectors such as agriculture, manufacturing, and logistics have been transformed and advanced by the emergence of robotics and autonomous systems in the past few decades. Unlike human who tends to make errors when carrying out tedious tasks over a long period of time, robots can perform such tasks efficiently. Tasks that may pose dangers to human, such as those dealing with hazardous substances or executed in harsh environments, can also be performed by robots.

Robots typically have some access to computational resources, which can be embedded in the robots themselves or located in a centralized server or in the cloud. These resources can be utilized to decide how the mission is going to be accomplished, which can be subject to some optimization criteria such as minimizing the time required to complete all tasks, minimizing the fuel usage, maximizing the usage of available resources, attempting to balance the workload, and many more.

With the steady research progress in artificial intelligence, robots are now able to perform various types of complex tasks, not just simple repetitive tasks as in the past. Based on the level of interaction required to accomplish the tasks, complex tasks can be classified into 2 types: 1) loosely-coordinated tasks, which are tasks that can be decomposed into several sub-tasks executed independently by different robots, and 2) tightly-coordinated tasks, which cannot be decomposed and require strong coordination between robots during execution (Rizk, Awad et al. 2019).

Multi-Robot System (MRS) is similar with Multi-Agent System (MAS), with the exception that agents in MRS are physical robots that can interact in physical space, whereas agents in MAS are distributed stationary computers (Yan, Jouandeau et al. 2013).

Based on their capabilities and properties, robots in MRS can be either homogeneous or heterogeneous. In homogeneous MRS, all the robots are equipped with identical actuators and sensing capabilities, whereas in heterogeneous MRS, each of them would have a unique set of capabilities and properties, making each robot suitable to handle specific types of tasks. In heterogeneous MRS, each task needs to be allocated to a robot that can fulfill the requirements of the task. MRS system assigns the tasks and schedules them in such a way that the overall execution cost is optimized.

When there is a massive number of tasks or when the tasks are complex, the system needs to apply task decomposition and task allocation. There are three different approaches known: decompose-then-allocate, allocate-then-decompose, or simultaneous allocation and decomposition approach. In decompose-then-allocate approach, task decomposition takes place before task allocation, allowing the task allocation process to be focused in optimizing a smaller number of tasks or a specific type of tasks. However, the resulted plan would be suboptimal since the task decomposition would not perform well without considering how the tasks are going to be allocated to the multiple robots. In allocate-then-decompose approach, task allocation takes place before task decomposition. This approach also produces a suboptimal plan because of the uncertainty of the costs of the allocated tasks, which are not exactly known before applying task decomposition. In simultaneous task allocation and decomposition, the robots simultaneously decompose the tasks into sub-tasks and trade the tasks or sub-tasks with one another (Zlot and Stentz 2006).

In this paper, we presented the architecture of heterogeneous MRS that performs task decomposition and task allocation to handle loosely-coordinated tasks, with the ability to perform task re-allocation in response to changing situation. We applied several algorithms, such as path finding and path optimization which involves heuristics and population-based metaheuristics for task allocation and re-allocation.

2 Problem definition

There are two types of missions we considered in this work. The first type of mission is a coverage problem, which is often found in scenarios requiring surveillance and target detection. As illustrated in Figure 1(a), the robots in a coverage problem are required to cover the whole area of a bounded region, where they perform certain sensing tasks or actions along the way. The second type of mission is route optimization, where the robots are required to follow a sequence of sensing tasks or actions at known locations as illustrated in Figure 1(b). Some examples of applications requiring a combination of these two missions are search-and-prosecute (Abukhalil, Patil et al. 2016) and disaster relief (Gregory, Fink et al. 2016).

Figure 1 Coverage and route optimization missions

In real-world scenarios, where the missions we need to solve are often complex in nature, heterogeneous robots with unique sets of capabilities are often more preferred than having one type of robot performing everything. Heterogeneous robots are usually more compact, cheaper, and easier to maintain.

In our work, we developed two types of robot prototypes, which are Unmanned Ground Vehicles (UGVs) and Unmanned Aerial Vehicles (UAVs). By having a wider angle from a higher altitude, UAVs can cover a wide area more effectively than UGVs, which is an essential feature for surveillance and target detection. On the other hand, UGVs can perform specific tasks on the ground that cannot be performed by UAVs. This illustrates how heterogeneous robots with different capabilities can complement each other.

When performing a mission, robots must not come into the perimeters of objects and areas which can cause harm to the objects or the robots. Those objects and areas, which are considered obstacles, could be of different types for different types of robots. For example, water bodies must be avoided by the UGVs, but not by the UAVs.

Locations of static obstacles can be loaded from a predefined map, if there is any, whereas the locations of moving obstacles such as human, animals, and other objects that are not available on the predefined map must be detected using real-

time sensors. Obstacles that can only be detected in real-time are one of the causes why task re-allocation is needed.

We identified some possible scenarios that may cause deviation from the initial plan, at which the task re-allocation needs to take place. In a dynamic environment, new tasks or targets might emerge at any point of time, causing the initial plan to be outdated. When this occurs, the initial plan needs to be updated to include the new targets. Besides, during mission execution, there is a chance that a robot has malfunctioned or incapacitated due to unforeseen circumstances. If the robot has not completed its assigned tasks, the remaining tasks need to be transferred and distributed to other robots. In the less extreme case, some robots could be performing slower than expected due to various internal and external factors, such as hardware deteriorations, incorrect predictions of task difficulties, obstacles detected along the way, and disturbances from the environment.

3 System architecture

We designed our heterogeneous MRS with a modular architecture, where the communication between each module is implemented using Robotic Operating System (ROS). As shown in Figure 2, there are 3 key modules under ROS topic that are connected to the ROS master: mission description, mission planner, and hardware coordinator.

Figure 2 System Architecture

Mission description provides a Graphical User Interface (GUI) that allows a human user to define a mission and trigger the task allocation as well as the execution of the plan. When the mission is being executed, the user can monitor the execution progress and be alerted when there is any condition that might demand a task re-allocation.

Mission planner generates a plan based on the mission requirements that has been defined by the user via GUI. It performs two stages of computation: path planning and optimization. During the path planning stage, mission planner computes the distances between every pair of points while considering obstacles that might present between the points. The distance values are stored and ready to be used by the optimizer. Since there are different types of robots, we need a separate cost computation for each type of robot. Then, the optimizer finds the allocation of the tasks and the sequences that minimize the total cost. The cost can be a single-objective cost, where the total distance or execution time is the only factor, or a multi-objective cost concerning other factors such as fuel consumption or load balancing.

Hardware coordinator encapsulates the robot modules in such a way that every robot behaves similarly from the point of view of mission planner. Robots can be emulated using simulator or run on the real hardware. Due to safety considerations of the robot hardware, we allow users to choose manual control and autonomous mode anytime during mission execution. In case the autonomous mode goes out of control, the users can take over the robot control using manual mode. We use Ardupilot open source autopilot firmware to enable autonomous control for both UAVs and UGVs, where the firmware parameters of the robots can be easily configured with a user-friendly interface.

4 Mission planning

In real-life missions, producing suboptimal plans are often considered satisfactory and more practical than attempting to get a global optimal plan due to several reasons. Firstly, relaxing the optimality of the plan would save a significant amount of optimization time and computational resources, especially when the scale of the mission is massive. Secondly, the cost of executing the global optimal plan might only be slightly lower than the cost of executing a suboptimal plan. The insignificant cost reduction may not be worth the extra time needed for planning. Lastly, mission execution may not go as planned due to unforeseen circumstances, particularly because the conditions encountered at the time of execution might have changed from the conditions captured at the time of planning. If the plan is most likely going to be amended due to the dynamic nature of the mission, it is not necessary to have a global optimal plan to begin with.

Task allocation is a combinatorial problem that falls into the category of NP-hard problems. To solve NP-hard problems, exact algorithms are rarely used because of their limited scalability for large problems. Heuristics methods are fast and scalable, but they tend to be quickly trapped at local minima. Evolutionary algorithms or metaheuristics, on the other hand, are known for their ability to produce reasonable solutions within a much shorter time as compared to exact algorithms. Some examples of metaheuristics are Genetic Algorithm (GA) (Holland 1992), Particle Swarm Optimization (PSO) (Shi 2001), Ant Colony Optimization (ACO)

(Dorigo and Blum 2005), and Simulated Annealing (SA) (Dowsland and Thompson 2012).

Unlike exact algorithms and heuristics, search process in metaheuristics always involves some stochastic randomness. Typically, the process involves generating an initial population and repeatedly applying evolutionary mechanisms such as crossover and mutation operators. The randomness introduces diversity into the solution candidates, which is essentially the principal mechanism that prevents the final solution from being trapped at local minima. However, relying primarily on a random-search mechanism is time-consuming and might not produce the desired optimal solution, especially for large problems where the size of solution space is enormous. As such, the incorporation of information either at the onset or progressively can enhance the capability of the search process.

Considering that we need to get a good solution quickly to fulfill the real-time requirement of the system, we use a Memetic Algorithm (MA) approach (Ong, Lim et al. 2010) which contains metaheuristics guided by memes. We use GA as the metaheuristics platform in this work. Since metaheuristics produce valid solutions from the beginning of the search, we can terminate the optimization process anytime according to the time budget that we have. Meanwhile, the search guidance given by the memes allow the solutions to improve faster at the beginning of the search, resulting in a good solution that we can use as early as possible.

As for the mid-mission task re-allocation, we implemented a local segment optimization that selectively reoptimize path segments when there is changing situation during mission execution. Selective re-optimization is much faster than re-optimizing the whole plan. Furthermore, for most practical scenarios, drastic or significant changes from the initial plan is undesirable or impractical.

Computational time of task re-allocation during mission execution is more critical than during initial task allocation because of the real-time nature of the mission execution. If the computation of the new plan takes longer time than the mission execution, the new plan cannot be distributed to the respective robots in time and the conditions would have changed from the time we initiated the task re-allocation. When this happens, the new plan we have generated would not be relevant anymore.

5 Conclusions

We have presented an architecture of a heterogeneous multi-robot mission planner system that is designed to support heterogeneous multi-robot, where there are several types of robots having unique sets of capabilities and properties. The main challenge we encountered is to optimize the task allocation while following the real-time requirements of the system. Therefore, we need to carefully select algorithms that can scale well with the problem size and provide good solutions within reasonable time.

References

- Abukhalil, T., et al. (2016). Coordinating a heterogeneous robot swarm using Robot Utility-based Task Assignment (RUTA). 2016 IEEE 14th International Workshop on Advanced Motion Control (AMC), IEEE.
- Dorigo, M. and C. Blum (2005). "Ant colony optimization theory: A survey." Theoretical computer science **344**(2-3): 243-278.
- Dowsland, K. A. and J. M. Thompson (2012). "Simulated annealing." Handbook of natural computing: 1623-1655.
- Gregory, J., et al. (2016). Application of multi-robot systems to disaster-relief scenarios with limited communication. Field and Service Robotics, Springer.
- Holland, J. H. (1992). "Genetic algorithms." Scientific american **267**(1): 66-73.
- Ong, Y.-S., et al. (2010). "Memetic computation—Past, present & future." IEEE Computational Intelligence Magazine **5**(2): 24-31.
- Rizk, Y., et al. (2019). "Cooperative Heterogeneous Multi-Robot Systems: A Survey." ACM Computing Surveys (CSUR) **52**(2): 29.
- Shi, Y. (2001). Particle swarm optimization: Developments, applications and resources. Proceedings of the 2001 Congress on Evolutionary Computation, IEEE.
- Yan, Z., et al. (2013). "A survey and analysis of multi-robot coordination." International Journal of Advanced Robotic Systems **10**(12): 399.
- Zlot, R. and A. Stentz (2006). "Market-based multirobot coordination for complex tasks." The International Journal of Robotics Research **25**(1): 73-101.