Algorithms for Multi-Robot Trajectory Planning in Well-formed Infrastructures

Michal Čáp
ATG, Dept. of Computer Science,
FEL, CTU in Prague, Czech Republic

Introduction

In recent years, the successful demonstrations of production-quality mobile robots, autonomous UAVs, and self-driving cars fueled excitement about the future opportunities offered by autonomous multi-vehicle systems both for transportation of goods and people. However, the efficiency and safety of such systems will depend on the existence of guaranteed methods for reliable collision avoidance between the individual vehicles. Unfortunately, the general formulation of multi-vehicle coordination problem has been shown to be PSPACE-hard (Hopcroft, Schwartz, and Sharir 1984) and thus the existing complete algorithms suffer from exponential worst-case time complexity in the number of coordinated robots. Thus, for a practical solution one has to resort to some heuristic method such as prioritized planning, which however may fail to provide a collision-free solution for some problem instances. Within our work, we propose a variation of the classical prioritized planning scheme called "Revised Prioritized Planning" and a characterize the class of instances that are provably solvable by the newly proposed algorithm. Further, we show that the algorithm can be adapted and executed in an asynchronous decentralized manner, such that each robot computes its own trajectory and individual robots arrive to a solution by exchanging messages. The idea can be also extended to a dynamic variant of the problem where the individual robots may be ordered to travel to particular destination at any time-point during the operation of the system.

Related Work

As an example autonomous multi-vehicle system, consider a future factory where intermediate products are moved between workplaces by autonomous robots. The worker at a particular workplace calls a robot, puts an object to a basket mounted on the robot and orders the robot to autonomously deliver the object to another workspace where the object will be retrieved by a different worker. Clearly, an important requirement on such a system is that each robot must be able to avoid collisions with other robots autonomously operating in the shared space. The problem of avoiding collisions between individual robots can be approached either from a control engineering perspective by employing reactive collision avoidance or from AI perspective by planning coordinated trajectories for the robots.

In the reactive approach, the robot follows the shortest path to its current destination and attempts to resolve collision situations as they appear. Each robot periodically observes positions and velocities of other robots in its neighborhood. If there is a potential future collision, the robot attempts to avert the collision by adjusting its immediate heading and velocity. A number of methods have been proposed (Van den Berg, Lin, and Manocha 2008; Guy et al. 2009; Lalish and Morgansen 2012) that prescribe how to compute such collision-avoiding velocity in a reciprocal multi-robot setting, with the most prominent one being ORCA (Van Den Berg et al. 2011). These approaches are widely used in practice thanks to their computational efficiency – a collision-avoiding velocity for a robot can be computed in a fraction of a millisecond (Van Den Berg et al. 2011). However, these approaches resolve collisions only locally and thus they cannot guarantee that the resulting motion will be deadlock-free and that the robot will always reach its destination.

With a planning approach, the system first searches for a set of globally coordinated collision-free trajectories from the start position to the destination of each robot. After the planning has finished, the robots start following their respective trajectories. If robots are executing the resulting joint plan precisely (or within some predefined tolerance), it is guaranteed that the robots will reach their destination while avoiding collisions with other robots. It is however known that the problem of finding coordinated trajectories for a number of mobile objects from the given start configurations to the given goal configurations is intractable. More precisely, the coordination of disks amidst polygonal obstacles is NP-hard (Spirakis and Yap 1984) and the coordination of rectangles in a bounded room is PSPACE-hard (Hopcroft, Schwartz, and Sharir 1984).

Even though the problem is relatively straightforward to formulate as a planning problem in the Cartesian product of the state spaces of the individual robots, the solutions can be very difficult to find using standard heuristic search techniques as the joint state-space grows exponentially with increasing number of robots. The complexity can be partly mitigated using techniques such as ID (Standley 2010) or

M* (Wagner and Choset 2015) that solve independent sub-conflicts separately, but each such sub-conflict can still be prohibitively large to solve because the time complexity of the planning is still exponential in the number of robots involved in the sub-conflict.

Instead, heuristic approaches are often used in practice, such as prioritized planning (Erdmann and Lozano-Pérez 1987), where the robots are ordered into a sequence and plan one-by-one such that each robot avoids collisions with the higher-priority robots. This greedy approach tends to perform well in uncluttered environments, but it is in general incomplete and often fails in complex environments.

When the geometric constraints are ignored, complete polynomial algorithms can be designed such as Push&Rotate (de Wilde, ter Mors, and Witteveen 2013) or Bibox (Surynek 2009). These algorithms solve so-called "Pebble motion" problem, in which pebbles(robots) move on a given graph such that each pebble occupies exactly one vertex and no two pebbles can occupy the same vertex or travel on the same edge during one timestep. Although this model can be useful for coordination of identical robots on coarse graphs[1], it is not applicable for trajectory coordination of robots with fine-grained or otherwise rich motion models.

Based on the previous discussion, we may see that when we are facing a problem of choosing a suitable method for coordinating the trajectories of individual robots in the system, we are forced to sacrifice either the tractability, completeness or the ability to model the collisions geometrically.

Problem Statement

Consider n circular robots operating in a 2-d workspace $\mathcal{W} \subseteq \mathbb{R}^2$. The maximum speed the robot i can move at is denoted as v_i. Each robot is assumed to be assigned a *task* that involves moving from its start position s_i to some goal position g_i and stay there. We assume that the start and goal positions of all robots are mutually disjunct, i.e. the bodies of robots do not overlap when the robots are on their start positions and when they are on their goal positions. The trajectories π_i, π_j of two robots i, j are said to be *conflict-free* if the bodies of the robots i, j never intersect when they follow the trajectories π_i and π_j. Then, we are interested in solving the following problem:

Problem. Given a workspace $\mathcal{W}$ and tasks $\langle s_1, g_1\rangle, \ldots, \langle s_n, g_n\rangle$ for robots $1, \ldots, n$, find trajectories $\pi_1, \ldots, \pi_n$ such that each trajectory π_i reaches the robot's destination g_i and trajectories π_i, π_j of every two different robots i, j are mutually conflict-free.

Also, in practical deployment scenarios, it is often advantageous if a) the algorithm can be run in a decentralized manner and b) if the algorithm can be used to coordinate trajectories in a running system, i.e. a new task for a robot may appear when other robots are on their way to their previously assigned destinations.

[1]The graph must be coarse enough so that the bodies of two robots "sitting" on two different vertices will never overlap. The same has to hold for two robots traveling on different edges of the graph.

Revised Prioritized Planning

A pragmatic approach that is often useful even for large multi-robot teams is prioritized planning. The idea has been first articulated by Erdman and Lozano-Pérez in (Erdmann and Lozano-Pérez 1987). Other works such as (van den Berg and Overmars 2005; Bennewitz, Burgard, and Thrun 2002) investigate techniques for choosing a good prioritization for the robots. In prioritized planning each robot is assigned a unique priority. The trajectories for individual robots are then planned sequentially from the highest priority robot to the lowest priority one. For each robot a trajectory is planned that avoids both the static obstacles in the environment and the higher-priority robots moving along the trajectories planned in the previous iterations. Prioritized planning is in general incomplete, consider the counter-example (Silver 2005) depicted in Figure 1:

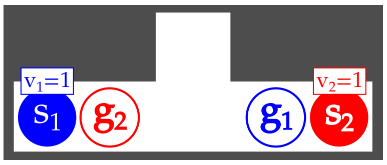


Figure 1: The picture shows two robots desiring to move from s_1 to g_1 (s_2 to g_2 resp.) in a corridor that is only slightly wider than a body of a single robot. The scenario assumes that both robots have identical maximum speeds. Observe that irrespective of which robot starts planning first, its trajectory will be in conflict with all satisfying trajectories of the robot that plans second.

Let us now analyze when is prioritized planning bound to fail. The algorithm fails to find a trajectory for robot i if 1) no satisfying path exists for robot i, i.e. the robot cannot reach its destination even if there are no other robots in the workspace; 2) every trajectory of robot i is in conflict with some higher-priority robot. There are two types of conflicts that can occur between a trajectory π of robot i and a higher-priority robot:

Type A: Occurs if trajectory π is in conflict with a higher-priority robot who has reached and is "sitting" at its destination, i.e. it is blocked by a static higher-priority robot.

Type B: Occurs if trajectory π of robot i is in conflict with a higher-priority robot who is moving towards its destination, i.e. it is "run over" by a moving higher-priority robot.

A question that naturally arises is whether it would be possible to restrict the class of solvable instances or to alter the prioritized planning algorithm such that there will always be at least one trajectory without neither Type A nor Type B conflict for each robot.

One way to ensure that there will be a satisfying trajectory without Type A conflict for every robot is to only consider instances, where each robot has a path to its goal that avoids goal regions of all higher-priority robots. When each robot follows such a path, then they cannot be engaged in a Type

A conflict, because a Type A conflict can only occur at the goal region of one of the higher-priority robots.

Unfortunately, the existence of a trajectory without Type B conflict is difficult to guarantee in classical prioritized planning, since higher-priority robots completely ignore interactions with lower-priority robots when planning their trajectories. To ensure that each robot will have a satisfying trajectory without Type B conflict, all higher-priority robots would have to plan their trajectories so that the lower-priority robots are always left with some alternative trajectory that can be used to avoid the potential conflicts of this type.

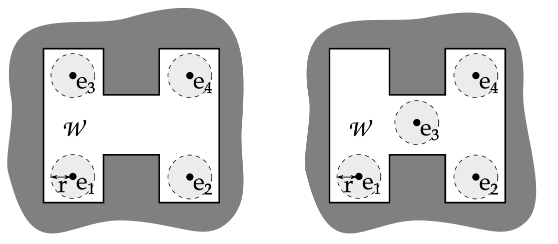
One way to ensure that there will be a satisfying trajectory without Type B conflict for every robot is to consider only instances where each robot has an option to follow a path to its goal that avoids start region of lower-priority robots and enforce that the trajectory of each robot will avoid start regions of all lower-priority robots. When this is ensured, then any robot will always have a fall-back option to wait at its start position (since no higher-priority robot can run over its start region) until its desired path is clear of all higher-priority robots. Thus it can always avoid Type B conflicts. Moreover, if the robot continues by following a path that avoids goal regions of higher-priority robots, then the resulting trajectory is also guaranteed to avoid the Type A conflicts.

We propose a Revised version of Prioritized Planning (RPP) (Čáp et al. 2014) that uses the insights from the preceding discussion and plans the trajectory of each robot so that both a) start position of all lower-priority robots are avoided and b) collisions with higher-priority robots are avoided. It can be shown that for trajectory coordination problems occurring in appropriately designed environments called *well-formed infrastructures*, the RPP algorithm is guaranteed to provide a solution (Čáp et al. 2014).

Well-formed Infrastructures

To model systems such as factories, rail roads, road networks etc., we introduce a notion of infrastructure. An infrastructure is a pair $(\mathcal{W}, E)$, where $\mathcal{W}$ is a workspace (described by a set of obstacle-free coordinates $\mathcal{W} \subseteq \mathbb{R}^2$) and a set of points $E \subset \mathcal{W}$ represents distinguished locations in the environment called endpoints (modeling e.g. workplaces in a factory, parking places, road stops, etc.). Vehicles operating in such an infrastructure can be assigned *relocation tasks* denoted $s \to g$ requesting the chosen robot to move from its current position $s \in E$ to the given goal endpoint $g \in E$, i.e. the vehicles moves only between the endpoints of the infrastructure.

A *well-formed* infrastructure has its endpoints distributed in such a way that any robot standing on an endpoint cannot completely prevent other robots from moving between any other two endpoints. In a well-formed infrastructure, a robot is always able to find a collision-free trajectory to any other unoccupied endpoint by waiting for other robots to reach their destination endpoint, and then by following a path around the occupied endpoints, which is in a well-formed infrastructure guaranteed to exist.



(a) Well-formed infrastructure: The workspace $\mathcal{W}$ and endpoints $\{e_1, e_2, e_3, e_4\}$ for robots having radius r form a well-formed infrastructure.

(b) Ill-formed Infrastructure: The workspace $\mathcal{W}$ and endpoints $\{e_1, e_2, e_3\}$ do not form a well-formed infrastructure because there is no path from e_1 to e_2 with $2r$-clearance to e_3 for a robot having radius r.

Figure 2: Well-formed and ill-formed infrastructure

In the following, we will describe the idea more formally. First, let us introduce the necessary notation. Let $D(x, r)$ be a closed disk centered at x with radius r. Then, $\operatorname{int}_r X := \{x : D(x, r) \subseteq X\}$ is an r-interior of a set $X \subseteq \mathbb{R}^2$ and $\operatorname{ext}_r X := \bigcup_{x \in X} D(x, r)$ is an r-exterior of a set $X \in \mathbb{R}^2$.

Definition 1. An infrastructure $(\mathcal{W}, E)$ is called *well-formed* for circular robots having body radii $r_1, \ldots, r_n$ if any two endpoints $a, b \in E$ can be connected by a path in workspace $\operatorname{int}_{\bar{r}} \left(\mathcal{W} \setminus \bigcup_{e \in E \setminus \{a,b\}} D(e, \bar{r}) \right)$, where $\bar{r} = \max\{r_1, \ldots, r_n\}$.

In other words, there must exists a path between any two endpoints with at least $\bar{r}$-clearance with respect to the static obstacles and at least $2\bar{r}$-clearance to any other endpoint. Figure 2 illustrates the concept of a well-formed infrastructure.

The notion of well-formed infrastructures follows the structure typically witnessed in man-made environments that are intuitively designed to allow efficient transit of multiple people or vehicles. In such environments, the endpoint locations where people or vehicles are stopping for longer time are separated from the transit area that is reserved for travel between these locations.

If we take the road network as an example, the endpoints would be the parking places and the system of roads is built in such a way that any two parking places are reachable without crossing any other parking place. Similar structure can be witnessed in offices and factories. The endpoints would be all locations, where people may need to spend longer periods of time, e.g. surroundings of the work desks or machines. As we know from our every day experience, work desks and machines are typically given enough free room around them so that a person working at a desk or a machine does not obstruct people moving between other desks or machines. We can see that real-world environments are indeed often designed as well-formed infrastructures and in such environments RPP can be used to find coordinated collision-free trajectories for the robots' relocation tasks in a guaranteed manner.

Asynchronous Decentralized Prioritized Planning

Consider a multi-robot system consisting of a large number of heterogeneous autonomous robots. In such a scenario, a decentralized implementation of (revised) prioritized planning may be more desirable than a centralized one. In a decentralized implementation, each robot runs its own instance of the algorithm and exchanges messages with the other robots according to a prescribed communication protocol. If an inconsistency is detected by a robot, then it recomputes the best trajectory for itself using its own on-board computation resources. The process should eventually converge to a state where all robots hold mutually conflict-free trajectories.

An advantage of such an approach is that several robots often end up computing their trajectories in parallel and thus a conflict-free solution is often computed faster. Another advantage for multi-robot systems with heterogeneous robots is that the kinematic and other potentially implicit constraints on the trajectory of a particular robot stay local to that robot and do not need to be formalized nor communicated, which simplifies the design of the communication protocol and allows each robot to use a custom robot-specific planner for planning its trajectory. We propose an asynchronous decentralized version of (revised) prioritized planning (ADPP/ADRPP) that works as follows:

When the computation is started, each robot computes an individually optimal trajectory to its desired destination and broadcasts the computed trajectory to all other robots. Each robot in the systems listens for those trajectory update messages and stores them locally to maintain an overview of the other robots' intended trajectories. Every time a new trajectory update is received a robot checks whether its current trajectory is collision-free with the intended trajectories of higher-priority robots and if needed replans to find a new collision-free trajectory that avoids collisions with the higher-priority robots. The new trajectory is again communicated to all other robots.

We show (Čáp et al. 2014) that such a decentralized approach inherits properties of both classical prioritized planning and revised prioritized planning – in particular it is guaranteed to terminate in finite time and the revised variant is guaranteed to always terminate with collision-free trajectories if presented with a coordinating problem in a well-formed infrastructure. Moreover, the decentralized approach often leads to a significant speed-up of the computation compared to the centralized approach – in our experiments, the algorithm achieved average 4x speed-up on instances with 20 robots.

Continuous Best-Response

The above-discussed algorithms (PP, RPP, ADPP and ADRPP) address the static formulation of the trajectory coordination problem, where the destinations of *all* robots are known apriori and the joint solution is computed and executed at once by all robots. However, this may be impractical in systems where the relocation tasks are assigned to the individual robots on-line (i.e. at any time-point) during the operation of the multi-robot system. Using one of the static algorithms in such a scenario would require to interrupt execution of the previously planned trajectories each time a new task is assigned, include the new task into the coordination problem and replan all trajectories in the system. However, when the execution of coordinated trajectories is interrupted half-way, the robots' current position will be in general outside of an endpoint and in such situations RPP/ADRPP algorithm is not guaranteed to provide a collision-free solution.

We propose a variation of the prioritized planning approach called continuous best-response approach (COBRA) (Čáp, Vokřínek, and Kleiner 2015) that addresses this problem. The newly proposed algorithm COBRA is a decentralized approach that employs a data token (Ghosh 2010) passed between the robots to synchronize computation and ensure that each robot works with up-to-date information about the trajectories of other robots. We identify a token Φ with a set $\{(a_i, \pi_i)\}$ that contains at most one tuple for each robot $a = 1 \ldots, n$. Each such tuple represents the fact that robot a is intending to move along trajectory π. At any given time the token can be held by only one of the robots and only this robot can read and change its content. Then, the algorithm works as follows:

A robot newly added to the system tries to obtain the token and to register itself with a trajectory that stays at its initial position forever. After all the robots have been added to the system, the user can start with assigning relocation tasks to individual robots:

When a new relocation task is received by a robot, the robot requests the token Φ. When the token is obtained, the robot runs a trajectory planner to find a new "best-response" trajectory to fulfill the relocation task. Such a trajectory is required a) to start at the robot's current position p at time $t_{\text{now}} + t_{\text{planning}}$ (at the end of the planning window), b) to reach the goal position g as soon as possible and remain at g and c) to avoid collisions with all other robots following trajectories specified in the token. If such a trajectory is successfully found, the token is updated with the newly generated trajectory and released so that other robots can acquire it. Then, the robot starts following the found trajectory. Once the robot successfully reaches the destination, it can accept new relocation tasks.

In (Čáp, Vokřínek, and Kleiner 2015), we show that if the relocation tasks assigned to the robots are between endpoints of a well-formed infrastructure, the trajectory planning of each robot is guaranteed to succeed and return a valid trajectory for the robot and task. Moreover, assuming that the trajectory planning is done on a discretized space-time lattice, the worst-case asymptotic time-complexity of finding a best-response trajectory is only quadratic in the number of robots operating in the system, contrasting exponential complexity of the complete algorithms for general formulation of the problem.

Conclusion

Reliably and tractable methods for trajectory coordination are important enablers for future autonomous multi-vehicle systems. However, the complete algorithms solving the general formulation of the problem have worst-case asymptotic time complexity that is exponential in the number of robots in the system. We have presented a family of algorithms for finding coordinate trajectories in multi-robot systems and characterized a class of problem instances (the coordination problems occurring in well-formed infrastructures) that are provably solvable by these algorithms in polynomial (more precisely quadratic) time. Different varaints of the approach were devised, suiting centralized, decentralized and on-line/dynamic multi-robot systems.

References

Bennewitz, M.; Burgard, W.; and Thrun, S. 2002. Finding and optimizing solvable priority schemes for decoupled path planning techniques for teams of mobile robots. *Robotics and Autonomous Systems* 41(2):89–99.

de Wilde, B.; ter Mors, A. W.; and Witteveen, C. 2013. Push and rotate: cooperative multi-agent path planning. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, 87–94. International Foundation for Autonomous Agents and Multiagent Systems.

Erdmann, M., and Lozano-Pérez, T. 1987. On multiple moving objects. *Algorithmica* 2:1419–1424.

Ghosh, S. 2010. *Distributed systems: an algorithmic approach*. CRC press.

Guy, S. J.; Chhugani, J.; Kim, C.; Satish, N.; Lin, M.; Manocha, D.; and Dubey, P. 2009. Clearpath: Highly parallel collision avoidance for multi-agent simulation. In *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '09, 177–187. New York, NY, USA: ACM.

Hopcroft, J.; Schwartz, J.; and Sharir, M. 1984. On the complexity of motion planning for multiple independent objects; pspace- hardness of the "warehouseman's problem". *The International Journal of Robotics Research* 3(4):76–88.

Lalish, E., and Morgansen, K. A. 2012. Distributed reactive collision avoidance. *Autonomous Robots* 32(3):207–226.

Silver, D. 2005. Cooperative pathfinding. In *AIIDE*, 117–122.

Spirakis, P. G., and Yap, C.-K. 1984. Strong np-hardness of moving many discs. *Inf. Process. Lett.* 19(1):55–59.

Standley, T. S. 2010. Finding optimal solutions to cooperative pathfinding problems. In Fox, M., and Poole, D., eds., *AAAI*. AAAI Press.

Surynek, P. 2009. A novel approach to path planning for multiple robots in bi-connected graphs. In *Proceedings of the 2009 IEEE international conference on Robotics and Automation*, ICRA'09, 928–934. Piscataway, NJ, USA: IEEE Press.

van den Berg, J., and Overmars, M. 2005. Prioritized motion planning for multiple robots. In *IROS*, 430–435.

Van Den Berg, J.; Guy, S.; Lin, M.; and Manocha, D. 2011. Reciprocal n-body collision avoidance. *Robotics Research* 3–19.

Van den Berg, J.; Lin, M.; and Manocha, D. 2008. Reciprocal velocity obstacles for real-time multi-agent navigation. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, 1928–1935. IEEE.

Čáp, M.; Novák, P.; Kleiner, A.; and Selecký, M. 2014. Prioritized planning algorithms for trajectory coordination of multiple mobile robots. *IEEE Transactions on Automation Science and Engineering (accepted).*

Čáp, M.; Vokřínek, J.; and Kleiner, A. 2015. Complete decentralized method for on-line multi-robot trajectory planning in valid infrastructures. In *International Conference on Planning and Scheduling, ICAPS 2015 (accepted).*

Wagner, G., and Choset, H. 2015. Subdimensional expansion for multirobot path planning. *Artificial Intelligence* 219:1 – 24.