2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

Path Negotiation for Self-interested Multirobot Vehicles in Shared Space

Hiroaki Inotsume[1,2]*, Aayush Aggarwal[1]*, Ryota Higa[1,2], and Shinji Nakadai[1,2]

***Abstract*— This paper addresses the problem of path negotiation among self-interested multirobot operators in shared space. In conventional multirobot path planning problems, most of the research thus far has focused on the coordination and planning of collision-free paths for multiple robots with some common objectives. On the contrary, the recent progress of technologies in autonomous vehicles, including automated guidance vehicles, unmanned aerial vehicles, and manned autonomous cars, has increased demand for solving coordination and conflict avoidance in these autonomous and self-interested agents that pursue their own objectives. In this research, we tackle this problem from the operator perspective. We assume a problem setting where collisions between robots are avoided based on path reservation and negotiation. Under that circumstance, we propose a task-oriented utility function and a path negotiation algorithm for robot operators to maximize their task utility during path negotiation. The simulation and experiment results demonstrate the effectiveness of our task-based negotiation method over a simple path-based negotiation approach.**

I. Introduction

Recent progress in the technological development and widespread use of unmanned aerial vehicles (UAVs), or drones, has increased the demand for the safe and efficient utilization of a shared airspace. Thus, traffic management systems and architectures have been widely discussed and developed worldwide [1], [2], most of which use the so-called first come, first served (FCFS) mechanism to avoid conflicts with UAVs and manned aircraft. Some of them have also been considering incorporating negotiation protocols into the traffic management framework where two or multiple UAV operators communicate their plans and negotiate for the airspace required for their missions. In addition, coordination and conflict avoidance between autonomous driving vehicles [3], [4], as well as between autonomous transport/delivery vehicles operating in shared spaces, are additional examples of the problem in which the vehicles or drivers have their own objectives, e.g., to arrive at a target location as early as possible or within a certain limited time. With these problems, all vehicles or vehicle operators act as a *self-interested agent* pursuing their own mission objectives and do not necessarily take cooperative actions with other agents unless the rules are posed.

Studies have been conducted in the fields of mechanism design and game theory to address conflicts between self-interested agents. For example, [5] proposed a combinatorial auction mechanism for solving multiagent path planning problems with self-interested agents, where the auction mechanism allocates paths to each agent such that the sum of the path costs of all agents is minimized. In addition, in [6], the authors proposed a negotiation mechanism in which a pair of two self-interested agents negotiates for conflict-resolving trajectories.

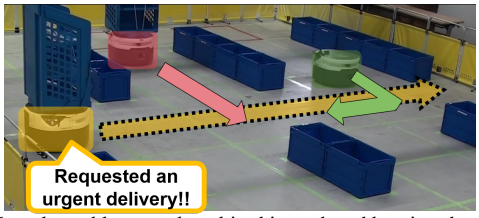


Fig. 1. Sample problem employed in this study, addressing the problem of path negotiation among multiple robotic vehicles working in a shared space with their own objectives and task deadlines.

For the path coordination, these studies assume that each agent evaluates its plan based on its path cost (e.g., length, duration, or energy usage). However, such a path-based evaluation is not always adequate to maximizing the profit of the agent. For example, when an agent's task has a deadline, it might not be met even if the agent can obtain a shorter path by negotiation if the agent only cares about reducing the path cost. By contrast, the agent might be able to successfully complete the task by evaluating its plan and negotiation deals based on a higher, task planning level.

In this study, we address the path negotiation problem among self-interested operators from an agent perspective (Fig. 1). We propose a task-oriented negotiation algorithm with which an operator evaluates its path and negotiated deals based on its task utility instead of each path cost. We conducted numerical simulations and physical experiments to evaluate the effectiveness of the proposed approach.

II. Related Studies

The problem of planning a set of conflict-free paths for multiple robots is generally called multirobot path planning (MPP) or multiagent path finding (MAPF) [7], [8]. Several algorithms for efficiently finding optimal or sub-optimal solutions have been proposed so far with centralized [9], [10] and decentralized [11]–[13] paradigms.

Thus far, the main focus of MPP–MAPF research has been on multiple robots within a team having a common mission, e.g., minimizing the sum or maximum travel duration of all robots. The main applications of these problems are autonomous transport vehicles in warehouses and logistical centers [14], [15]. In this study, by contrast, we address the problem in which there is more than one non-cooperative,

*The both authors equally contributed to this work.

[1]Data Science Research Laboratories, NEC Corp., 1753 Shimonumabe, Nakahara-ku, Kawasaki, Kanagawa, Japan {h-inotsume, aayush, r-higaryouta, nakadai}@nec.com

[2]Artificial Intelligence Research Center, National Institute of Advanced Industrial Science and Technology, 2-4-7 Aomi, Koto-ku, Tokyo, Japan

self-interested agent, each of which acts to maximize its own task utility without regard to the welfare of the other agents. In addition, we assume that their objective functions are not shared with the other agents.

Owing to this nature of self-interested agents, one of the main difficulties in a self-interested MPP problem is that we cannot utilize approaches that are widely used in a conventional single-team, cooperative MPP problem where a central system or a local leader agent can coordinate a set of path plans or decide which agent goes first based on their common objective. To deconflict self-interested agent motions, implicit coordination mechanisms have been proposed. For instance, in [16], the authors proposed tolling mechanisms that increase the cost of using areas through which multiple agents intend to pass. In [4], [5], auction mechanisms were introduced to allocate spatial resources to agents in which each agent bids on its preferable path.

In addition to these cost-adjustment-based approaches, negotiation-based deconfliction mechanisms have been investigated (e.g., [6], [17], [18]). In such negotiation mechanisms, agents communicate to one or more other agents regarding their actions to avoid conflicts. Among the studies conducted, in [6], a negotiation mechanism is proposed that guarantees the confidentiality of each agent's mission information, such as the destination and utility function, to a certain extent. In general, mechanisms and strategies for negotiations among multiple agents have been studied for decades in the field of automated negotiation [19], [20].

The main purpose of the above-mentioned research is to design mechanisms that can avoid conflict between self-interested agents. This study, by contrast, focuses on the problem of generating a better operation plan that improves the task utility of self-interested agents from each agent's perspective, given a negotiation mechanism. We consider a setting in which each agent negotiates with another agent to obtain a partially shared space for its task objective. The contributions of this study include the development of a path negotiation algorithm and a task-oriented metric for evaluating each negotiation. In addition, we implemented the proposed algorithm on test systems and performed experiments of path negotiations with physical robots. The experiments, along with numerical simulations, demonstrate the effectiveness of the proposed approach.

III. Problem Definition

In our problem, we are given K robot operators $o_1, \cdots, o_K$ in a shared space, with each operator owning a single robot r_k. Each robot r_k has its start location s_k. The operator o_k has its own task τ_k to complete. Each task is given to the operator at arbitrary times. The task τ_k comprises goal location g_k, deadline t_k, and reward r_k. The operator can gain the reward r_k only if the robot successfully completes the task (e.g., its robot reaches the goal g_k within the deadline t_k).

In our problem setting, each operator is self-interested; that is, it has its own mission objective and utility function that are not shared with others. Each operator attempts to maximize its task utility by completing its tasks. To do so, the operator should plan and assign a path to its robot from the robot's location s_k to the task goal g_k.

The entire shared space is divided into discrete spatiotemporal volumes with a two-dimensional space in addition to the time dimension. Subsequently, the set of discretized volumes is then encoded to a graph $G(V)$, where a vertex $v \in V$ represents a location at a specific time.

We consider discrete time steps $t \in T$, and denote v_k^t as a vertex robot r_k has occupied at time step t. At each time step, each robot r_k can either move from its current vertex v_k^t to its adjacent vertex v_k^{t+1} using the edge between (v_k^t, v_k^{t+1}) or wait at the same vertex v_k^t (i.e., $v_k^t = v_k^{t+1}$). A path for the robot r_k is a sequence of vertices that the robot occupies at each time step, as denoted by $P_k = (v_k^0, v_k^1, \cdots, v_k^T)$. No paths P_k of the robot r_k should be in conflict with any other paths $P_{k'}$ of the robot $r_{k'}$. Here, a path conflict can be either a vertex conflict (i.e., $v_k^t = v_{k'}^t$) or edge conflict (i.e., $(v_k^t = v_{k'}^{t+1}) \wedge (v_k^{t+1} = v_{k'}^t)$). In addition, a path cannot go through prohibited vertices (obstacles or no-go zones).

A. Reservation-based conflict avoidance

Conflicts between any two paths are avoided using a reservation-based resource allocation. We assume an area manager manages the entire shared space. The area manager has a reservation table that records the status of all vertices as either "free," "reserved," or "prohibited." Prohibited vertices denote locations that cannot be utilized by any operators either temporarily or perpetually. Reserved vertices are those already reserved and assigned to operators. Free vertices are volumes that are neither prohibited nor reserved and can be used by any operator. The reservation table tracks the status of every vertex in the graph G.

Each operator must submit a request for registering the paths for its mission. In this study, reservations are made based on a simple FCFS mechanism. The earliest valid path submitted is approved, and the corresponding vertices are reserved. Any other paths submitted after that are rejected if they conflict with the already reserved path. This corresponds to a decentralized version of the Cooperative A* algorithm [9]. Although only FCFS is considered in this study, note that the reservation mechanism is not limited to this; other possible mechanisms include auction-based space allocation [4], [5] and priority-based sequential planning [13]. The main idea of the negotiation approach proposed in this study can be applied to scenarios with other such mechanisms.

B. Path negotiation

We further assume that a negotiation mechanism is provided to all operators to enhance the flexibility and efficiency of the shared-space utilization. With negotiations, operators request or provide already reserved vertices. Within a negotiation, an operator sends a negotiation message $\mathbf{y}$ to another operator. The negotiation message contains requested vertices, providable vertices, the amount of payment, and the response. Requested vertices are those the message sender requests to the receiver. Providable vertices are, by contrast, those the sender has reserved and can provide to

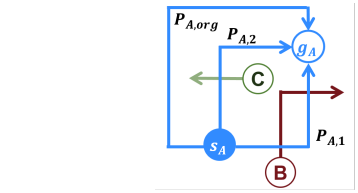
the receiver. Payment is the fee the sender can provide to the receiver for the requested vertices. The payment can be either positive, negative, or zero; a positive payment implies the sender pays a certain amount to the receiver, and a negative value indicates the opposite situation. This payment is introduced into the negotiation mechanism to induce an incentive for the requested operator to provide its reserved vertices. Otherwise, the provider might have no reason to concede such vertices if it has a self-interest and if no rule is imposed to foster a coordination.

During each negotiation step, a message receiver either accepts, conditionally accepts, or rejects the negotiation depending on the issue and its own mission. The receiver may reject or cancel the negotiation if some of its tasks become unattainable when the operator provides the requested vertices. In a conditional acceptance option, the receiver can set the requested and available vertices, and/or payment. The receiver creates a new response message and sends it to the other operator.

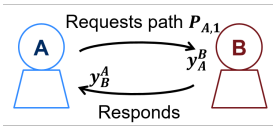
A negotiation continues until either a mutual agreement is made, one of the operators rejects or cancels the negotiation, or some predefined maximum negotiation count and/or time is reached. In the case of a mutual agreement, each operator involved in the negotiation submits the negotiation result to the area manager along with new paths the operator wants to reserve. The area manager then checks the consistency of the messages and confirms whether none of the newly submitted paths conflict with reserved paths. If inconsistency or conflict does not exist, then the area manager updates the reservation table to reflect the negotiation result.

Fig. 2 shows an example of the path negotiations. In this study, we consider simple bilateral negotiations, and an agent does not negotiate with more than one other agent to acquire a single path. That is, a path generated for a negotiation can conflict with only one agent's already reserved path. However, the agent can negotiate with multiple agents in parallel for different paths and decide which negotiation to apply to make a final agreement after processing all of these multiple negotiations. More complicated negotiation protocols may be adopted, such as multilateral and/or multiparty negotiations or sequential negotiations where a negotiated path collides with multiple agents' paths, or chained negotiations where a requested agent negotiates with a third agent for determining the deal with the first requesting agent. However, these complicated cases are out of the scope of the present study.

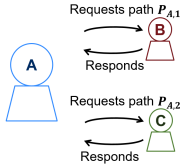
As another assumption, only one fixed path can be negotiated during a negotiation with each operator, and the amount of payment for that path is the only issue of the negotiation; that is, agents cannot change their requested paths during a negotiation with another agent. In a more strategic negotiation, a re-planning and updating of the negotiated path according to the previous response may also be possible. In this case, the negotiation will handle the pair (path and payment), but this is not considered herein.



(a) Example problem



(b) Single negotiation



(c) Parallel negotiations

Fig. 2. Example path negotiations. (a) Agent A plans a path from its start s_A to the task goal g_A. The optimal path without negotiation $P_{A,org}$ avoids conflicts with the already reserved paths of agents B and C. The paths $P_{A,1}$ and $P_{A,2}$ are shorter, but they conflict with the paths of B and C, respectively. (b) Agent A can negotiate for a single path with another agent or (c) with multiple agents for different paths in parallel.

C. Planning problem for robot operator

In the above setting, the problem for a robot operator is to plan a path for completing each task such that the operator maximizes its own task utility under the constraints of the available vertices and the task deadline. If the operator can find no such path for completing the task, then the operator may need to enter negotiations with another agent or execute the task without success or reward.

IV. Algorithms

In this section, we propose a task-oriented metric to evaluate a negotiation, along with a set of planning algorithms for a path negotiation for self-interested robot operators to maximize their task utility.

A. Negotiation utilities

During a negotiation, an operator should evaluate the deal based on a specific metric allowing the operator to make the most of the negotiation or avoid a disbenefit. Here let us assume that the negotiation is regarding a set of vertices V_{nego}. The requesting (buying) and requested (selling) operators evaluate how much value or utility V_{nego} has for them.

A straight forward way to do so is based on how much path length is reducible when the buying agent obtains V_{nego}, or how much extra path cost is required when the selling agent provides V_{nego}, as described in some previous studies. This conventional path-oriented utility metric can be expressed as

$$u_{nego,path}(V_{nego}) = c(P^*_{org}) - c(P^*_{new}) - p(V_{nego}), \qquad (1)$$

where $c(P)$ denotes the cost of the path P used in the operation. In addition, P^*_{new} and P^*_{org} are the optimal paths with and without negotiation. For a buyer P^*_{new} and P^*_{org} are the optimal paths that are generated with and without V_{nego}, respectively. For a seller they are the optimal paths without and with using V_{nego}, respectively. $p(V_{nego})$ denotes the payment for V_{nego}, and is normalized such that a unit payment becomes equivalent to the unit cost.

As described earlier the above path-oriented metric is not always adequate for maximizing a task profit. This study proposes the following task-oriented negotiation utility to evaluate a negotiation at a task level:

$$u_{nego,task}(V_{nego}) = u_{task}(\mathbf{x}^*_{new}) - u_{task}(\mathbf{x}^*_{org}) - p(V_{nego}), \quad (2)$$

where $u_{task}(\mathbf{x})$ denotes the utility function of the operation plan $\mathbf{x}$. $\mathbf{x}^*_{new}$ and $\mathbf{x}^*_{org}$ are the optimal task plans with and without the negotiation, respectively. Again for a buyer, $\mathbf{x}^*_{new}$ is the task plan that uses V_{nego} and $\mathbf{x}^*_{org}$ is the plan that does not use V_{nego}. For a seller they are the opposite.

Each operator requests or responds to a negotiation such that the negotiation utility satisfies the following condition:

$$u_{nego} > \varepsilon, \quad (3)$$

where ε denotes an operator-dependent threshold. Typically, ε is set to zero; thus, the operator can obtain some profit or at least not lose its utility through negotiations. The negotiation payment $p(V_{nego})$ is determined in such a way that Eq. (3) is met.

The task utility and cost functions $u_{task}(\mathbf{x})$ and $c(P)$ can be of any arbitrary form, and each operator can have its own function. For the problem described in Section III, we use the following task utility funcition for completing a task:

$$u_{task}(\mathbf{x}) = r(\mathbf{x}) - c(P). \quad (4)$$

The reward $r(\mathbf{x})$ becomes r_{max} if $\mathbf{x}$ can complete the task by the deadline; otherwise, zero. We used the duration of the path P as the path cost $c(P)$, although the cost function is not limited to this duration.

B. Planning and negotiation algorithm

An overview of a planning algorithm for the k-th operator o_k is shown in Algorithm 1. The operator first searches the optimal, conflict-free path P_{org} using only the free vertices (lines 1 and 2). The operator then plans negotiations with other operators to acquire a better solution. For each of the other operators, $o_{k'}$, o_k initially plans the optimal path P_{new} assuming that it can use the vertices reserved by $o_{k'}$ (lines 5 and 6). The operator then evaluates the newly planned path (line 8). If the path for negotiation is beneficial, then o_k initiates a negotiation with $o_{k'}$ under a zero payment (lines 9 and 10). Here, o_k can request only the conflicting vertices, but doing so may cause other conflicts with the alternative path of $o_{k'}$. Therefore, o_k should request the entire path or a set of vertices that sufficiently cover the path within the negotiation planning horizon. If the negotiation response from $o_{k'}$ is either "Agree" or "Deny," then o_k ends the negotiation with $o_{k'}$ (lines 11–14). If not, o_k evaluates the utility of the negotiation $u_{nego,k'}$ and determines the payment for the next negotiation iteration (lines 15 and 16). Here, o_k repeats this procedure until the iteration reaches the maximum number. Once the negotiations with all other operators are finished, o_k selects the best among them and sends a final agreement to the best negotiation partner o_{best}. In addition, o_k sends out cancellations of the negotiations to the other operators (lines 19–24).

Algorithm 1 Bilateral negotiation for operator o_k

1: $V_{use} \leftarrow V_{free}$
2: $P_{org} \leftarrow$ PlanPath(Robot,Task,V_{use})
3: $o_{best} \leftarrow NULL$; $u_{max} \leftarrow \varepsilon$;
4: **for** Each $o_{k'} \in \mathbf{O} \backslash o_k$ **do**
5: $V_{use} \leftarrow V_{free} \cup V^{k'}_{reserved}$
6: $\mathbf{P}_{new} \leftarrow$ PlanPath(Robot,Task,V_{use})
7: $i \leftarrow 0$; payment $\leftarrow 0$;
8: **while** $i <$ Maximum iteration count **and** NegotiationUtility(P_{org}, P_{new}, payment) $> \varepsilon$ **do**
9: $\mathbf{y}^{k'}_k \leftarrow$ SetNegotiationMessage($o_{k'}, P_{new}$,payment)
10: SendNegotiationRequest($o_{k'}, \mathbf{y}^{k'}_k$)
11: $\mathbf{y}^k_{k'} \leftarrow$ onReceivedNegotiationResponse($o_{k'}$)
12: **if** $\mathbf{y}^k_{k'}$.response = Agreement or Denial **then**
13: break
14: **end if**
15: $u_{nego,k'} \leftarrow$ NegotiationUtility($P_{org}, P_{new}, \mathbf{y}^k_{k'}$.payment)
16: payment $\leftarrow$ SetPayment($u_{nego,k'}$)
17: $i \leftarrow i+1$
18: **end while**
19: **if** $u_{nego,k'} > u_{max}$ **then**
20: $o_{best} \leftarrow o_{k'}$; $u_{max} \leftarrow u_{nego,k'}$; $\mathbf{y}_{best} \leftarrow \mathbf{y}^{k'}_k$;
21: **end if**
22: **end for**
23: SendNegotiationAgreement($o_{best}, \mathbf{y}_{best}$)
24: SendNegotiationCancellation($\mathbf{O} \backslash o_k, o_{best}$)

Algorithm 2 Responding to a negotiation request

1: $V_{use} \leftarrow V_{free} \cup V^{k'}_{reserved} \setminus V^k_{requested}$
2: $P_{new} \leftarrow$ PlanPath(Robot,Task,V_{use})
3: **if** $P_{new} = \emptyset$ **then**
4: $\mathbf{y}^k_{k'} \leftarrow$ SetNegotiationMessage(Denial)
5: **else**
6: $u_{nego,k} \leftarrow$ NegotiationUtility($P_{org}, P_{new}, \mathbf{y}^{k'}_k$.payment)
7: payment $\leftarrow$ SetPayment($u_{nego,k}$)
8: $\mathbf{y}^k_{k'} \leftarrow$ SetNegotiationMessage(o_k, P_{new}, payment)
9: **end if**
10: SendNegotiationResponse($o_k, \mathbf{y}^k_{k'}$)

Algorithm 2 describes the algorithm used by the operator $o_{k'}$ for not taking a loss during the negotiation with the requester o_k. Here, $o_{k'}$ first generates an alternative path $\mathbf{P}_{new}$ that avoids conflicts with the vertices requested by o_k (lines 1 and 2). If no collision-free path that can complete its task is found, $o_{k'}$ sends a denial to the negotiation (lines 4 and 5). Otherwise, $o_{k'}$ evaluates the new path $\mathbf{P}_{new}$, sets the payment for a request, and a responds to the negotiation with a conditional (lines 6–8).

The function PlanPath(Robot,Task,V_{use}) in Algorithms 1 and 2 searches for the optimal conflict-free path from the robot's current location to the task goal using the vertices V_{use}. In this study, we used the spatiotemporal A* algorithm to generate collision-free paths for a robot. As mentioned previously, when searching for the best path without a

negotiation, only the free vertices are set as available and reserved and prohibited vertices are treated as spatiotemporal obstacles. This guarantees that the solution will not collide with the plans of the other robots . During a negotiation, the path planner plans a new optimal path that uses additional vertices reserved by a negotiation partner $o_{k'}$. If the negotiation partner requests any vertices in the deal, then the planner excludes those requested vertices in the path re-planning.

The functions NegotiationUtility() and SetPayment() decide the negotiation policy, and different operators can have different policies. NegotiationUtility() calculates the utility of the negotiation based on either Eqs. (2) or (1). The payment amount and/or negotiation response are determined based on the negotiation utility such that condition Eq. (3) will be met. Examples of how to determine the payment are described in the next evaluation section.

V. Evaluation

To evaluate the proposed negotiation utility metric and planning algorithm, a set of numerical simulations were conducted. In addition, real-world experiments were performed with physical mobile robots.

A. Numerical simulations

Sets of simulation studies were conducted under various conditions to evaluate the effectiveness of the proposed algorithms.

In all simulation sets, the proposed algorithms and simulation modules are implemented in Python. For the negotiation protocol, *NegMAS* (Negotiation Multiagent Systems) [21] was integrated into the simulator. NegMAS implements various types of negotiation protocols. During this simulation, the stacked alternating offers protocol [22] was utilized to simulate bilateral negotiations between two agents. During the negotiation, both the requesting operators (buyers) and requested operators (sellers) use linear functions in SetPayment() to determine the payment at each negotiation iteration, i.e., buyers increase, and sellers decrease, the amount of payment by Δp at each iteration.

1) Simulation 1–varied agent and map sizes: During the first simulation set, the proposed algorithm was evaluated under various numbers of agents and map sizes. The agent size was varied from 20 to 100, and the map size was set to 5 x 5, 10 x 10, or 20 x 20 grids for each agent condition. During this simulation, 4-connected, no-obstacle maps were used. The simulation with each condition was run 20 times with randomly selected agent starting locations, task goal locations, and task deadlines, with and without a negotiation. The deadline for each task was specifically determined by adding a random integer of 0–10 to the shortest time required for the agent when moving from its start position to the goal. The negotiation utility of each agent was set to the proposed task-oriented function, as shown in Eq. (2). We then compared the results based on the following criteria: task success rate, task utility, path cost, negotiation rate, agreement rate, and run time. The task success rate represents the ratio of the number of agents that can complete the task by the deadline to the total number of agents. The negotiation rate represents how many negotiations occurred in a single run. The negotiation rate was normalized based on the total number of agents. The agreement rate represents the total number of agreements made divided by the total number of negotiations. The reward for every task was set to 100. During the simulations, we set each vehicle as a point-type agent; that is, it can move in any connected directions without changing its orientation. In addition, we assumed that each vehicle appears on the map only during the approved path duration. Each vehicle enters into the map at the path start time and disappears when it reaches its task goal. This corresponds somewhat to a UAV scenario in which the managed shared space is a certain height of airspace and UAVs only appear in the airspace after taking off and before landing.

We also tested our algorithm on a larger map with obstacles. For this purpose, one of the pathfinding benchmark maps [23], "den009d" shown on the left of Fig. 4 was used. The map has a grid size of 50 x 34 and a narrow corridor around its bottom center, which can induce congestion. In this case, we simulated a ground transport vehicle scenario in which, unlike the above UAV scenario, each vehicle remains on the map from the start to the end of the simulation.

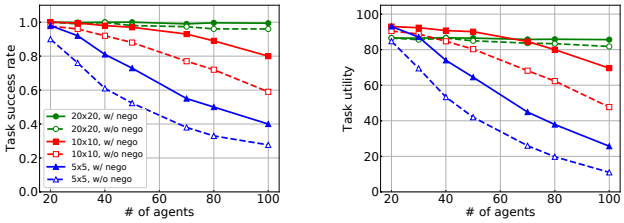
Both simulations were run on a machine with a 3.6-GHz Intel Core i7-7700 processor with 8 GB of RAM.

Fig. 3 shows the results of the first set of simulations. Several evaluation metrics were plotted against the varying number of agents for each map size. Note that the mean of the 20 trials is depicted on each plot. When comparing the task success rate and task utility on the 5 x 5 and 10 x 10 maps, planning with negotiations achieved a better result than that without negotiations, as shown in Figs. 3 (a) and (b). The success rate and utility improved by introducing negotiations. However, no significant difference can be observed in the mean path cost, as indicated in Fig. 3 (c). This occurs as a result of the increase and reduction in the path length, by which the negotiations cancel each other out because the same linear path cost function was utilized for all agents.

On the 20 x 20 map, no difference in the task success rate, task utility, or path cost between planning with or without negotiations can be seen. This is because the map is sufficiently large for the agents to reach their goals without taking long detours to avoid others. This can also be observed in Fig. 3 (d) where the negotiation rate decreases when the map size increases.

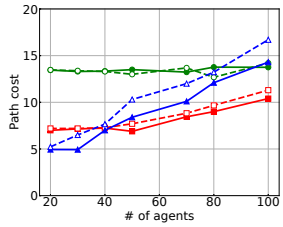
When comparing the results with regard to the agent size, the success rate and utility decrease with an increase in the number of agents even when negotiations are introduced. This can be attributed to a decrease in the agreement rate, as shown in Fig. 3 (e), for the 5 x 5 and 10 x 10 maps. Note that the number of agreements itself increases along with the increase in the agent size.

Finally, the runtime of the planning for all agents is shown in Fig. 3 (f). The runtime increases in a polynomial fashion with an increase in the agent size. This is because the k-th agent computes possible negotiation requests to k-1 prior

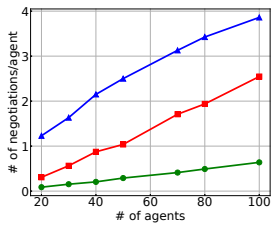


(a) Task success rate

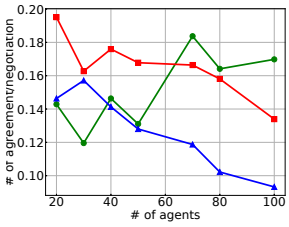
(b) Task utility



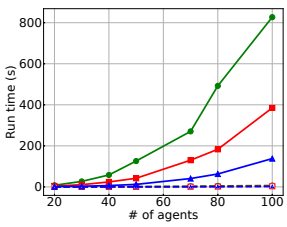
(c) Path cost



(d) # of negotiations/agent



(e) # of agreements/negotiation



(f) Run time

Fig. 3. Results of simulation set 1 with varied agent and map sizes. Evaluation metrics were plotted against the agent size.

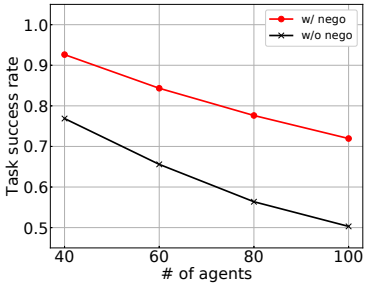


Fig. 4. Simulation results on the benchmark map. (Left) the den009d map. (Right) task success rate with and without negotiations with different numbers of agents.

agents, so the computational complexity is proportional to K^2 for the total of K agents.

Fig. 4 shows the result on the benchmark map. Here, only the task success rate is shown because the trends of all evaluation metrics are basically same as those in Fig. 3. As shown in Fig. 4, the task success rate can be improved using the proposed negotiation algorithm when compared to the cases without a negotiation, even with larger obstacle maps and under a ground vehicle scenario.

2) Simulation 2–task-oriented vs path-oriented negotiation: In this simulation set, we compared the proposed task-oriented negotiation with conventional path-oriented negotiation. We set the negotiation utility function of the requesting (buyer) and requested (seller) agents to either the task-

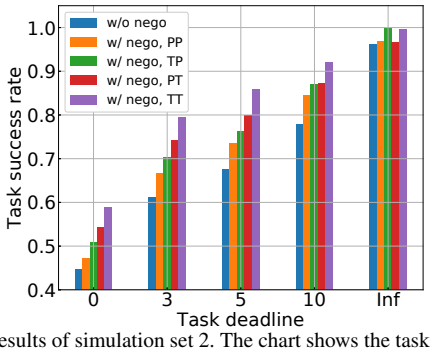


Fig. 5. Results of simulation set 2. The chart shows the task success rate for different combinations of negotiation utility for the buyer and seller agents. "TP", for example, represents buyers using the task-oriented negotiation utility function and sellers using the path-based function to evaluate a deal.

oriented (Eq.2) or path-oriented (Eq. 1) functions. We also varied the task deadline to 0, 3, 5, 10, and infinity. Deadline = 0 indicates that the deadline of all tasks equals the shortest path duration. Deadline = 3, 5, or 10 indicates that each task deadline is randomly set to the shortest path duration added by a random integer from zero to that number. Deadline = infinity indicates that no deadline is set for any of the tasks, and each task is treated as successfully completed whenever the vehicle reaches the task goal. We set the number of agents to 40 and the size of the map to 10 x 10. The simulations were run 20 times under each condition with randomly selected start and goal locations.

The task success rate with different combinations of negotiation utility are plotted in Fig. 5. In the legend, "T" and "P" represent the task-oriented and path-oriented negotiations, respectively, for buyers and sellers. For example, "TP" represents buyers using the task-oriented criterion to evaluate a negotiation deal, while sellers use the path-oriented criterion. The results without a negotiation are also plotted as a baseline.

The task success rate decreases when the task deadline becomes shorter for all cases. When comparing the different combinations of the negotiation utility functions, the success rate is the highest with the "TT' combination; that is, both buyers and sellers evaluate the deal based on the task-oriented utility function. Negotiation with the "PP" combination is the worst among the four, although it can still obtain better results than that without a negotiation. We can also notice that the "PT" setting is better than the "TP" setting. This is because, when a selling agent uses a path-based evaluation, it only considers the increase in the path length and does not regard the task deadline. This method for evaluating a deal can result in selling the reserved vertices for the successful completion of its task even though the alternative path does not lead the agent to the goal by the deadline. However, when a buyer uses the path-based evaluation metric, the agent can also possibly lose its chance to obtain a path for the task completion by underestimating the value of such a path, but it may still be able to obtain such a path by negotiating with multiple agents. When the deadline time is infinite, the differences in the task success rate are insignificant for all cases.

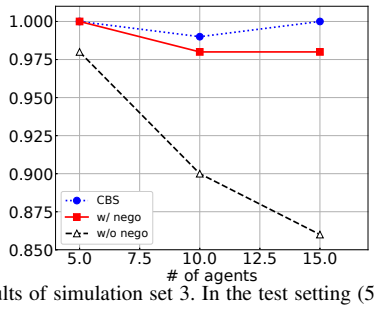


Fig. 6. Results of simulation set 3. In the test setting (5 x 5 map), the task success rates with a negotiation achieved values extremely close to those of the optimal CBS solutions.

The above results indicate the effectiveness of the proposed task-oriented negotiation approach over the path-based evaluation, especially when a task has a deadline. In the rest of the simulations and physical experiments, we used the proposed task-oriented negotiation approach for both buying and selling agents.

3) Simulation 3–comparison with centralized optimal solver: During the last simulation set, we compared our proposed algorithm with conflict-based search (CBS) [10], which is an optimal, centralized path assignment algorithm. We assumed that CBS was run by the central area manager and that the area manager knows all operator task utility functions for the optimization. The original CBS was designed to achieve an allocation such that the sum of all agent path costs is minimized. In this study, we modified the object function such that CBS maximizes the sum of all agent task utilities. Although CBS can find an optimal solution in a relatively efficient manner, finding such a solution is fundamentally NP-hard, and the problem becomes intractable when the number of agents increases. Therefore, in this simulation set, we tested using a smaller number of operators, 5, 10, and 15 and applied a map of 5 x 5 grids. The simulations were run on a machine with a 3.19-GHz Intel Core i7-8700 processor with 32 GB of RAM.

Fig. 6 shows the task success rate with a negotiation, along with the rates using a CBS and without a negotiation for the varied agent sizes. Note that, with 15 agents, CBS cannot find a solution within 30 min of computations in 2 out of the 10 trials. The plot shows the average values of the eight successful trials. Both with and without negotiations, the planning of all operators finished within less than 1 s. As shown in Fig. 6, CBS achieved the best task success rate among the three cases with a mean value of nearly 1. CBS allows multilateral negotiations for addressing all conflicts, whereas our algorithm only enables negotiations over those paths that have a single conflict. However, CBS requires the central area manager to know the task utility function of the operator to obtain the solution. Our algorithm does not require each operator to disclose its utility function, but still results in a task success rate extremely close to that of the optimal solution. Although the sizes of the agent and map are limited in this simulation set, the results indicate the possibility of the proposed negotiation algorithm generating a near optimal solution.

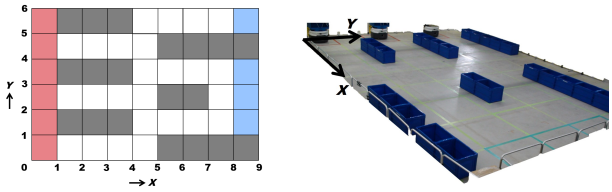


Fig. 7. Physical experiment setup with three mobile robots in 9 m x 6 m environment. Top: Map used during experiments. Bottom: Test environment.

B. Physical experiments

Physical experiments were conducted to evaluate the proposed approach using real robotic vehicles. During these experiments, we simulated multiple transport vehicles working in a shared workspace. Fig. 7 shows the experimental setup. During the experiments, three robot operators were introduced, with each operator owning one robotic vehicle. As a test map, a grid map of 9 m x 6 m with a 1 m x 1 m grid was used. The start location of each vehicle was fixed to one of the red areas (denoted as the delivery locations) depicted at the top of Fig. 7. The vehicle was assigned a set of tasks to pick up an item from the pick-up area (depicted in blue) and deliver it to the specified delivery location. Each pick-up or delivery task has its own reward and deadline.

Each task was given to one of the robot operators at different times. Therefore, planning and negotiation were conducted in an online approach, rather than in a one-shot manner as in the simulations described in the previous subsection. A newly task-provided operator planned a path and initiated a negotiation with the operator with the best possible negotiation utility. In this initial test campaign, the negotiation target operator temporarily stopped its robot motion, if it had already started executing its plan, to simplify the implementation. When the requested operator planned an alternative path to de-conflict the requested path of the requesting operator, it planned the path from the current robot location to the task goal. Owing to this online scenario, the negotiation policy was set such that each negotiation finished immediately. An operator initiated a negotiation with another operator if the negotiation could improve its task utility ($u_{nego,task} > 0$). When a negotiation was requested, the selling operator either denied the request if the best alternative path could not meet the task deadline, or conditionally agreed with the request along with the requested payment corresponding to the reduced task utility (equivalently increased path cost). The buying operator made an agreement if the requested payment still met $u_{nego,task} > 0$, or otherwise canceled the negotiation. This resulted in each negotiation ending within one round.

All of the experiment systems, including the planning and evaluation algorithms, were implemented in C++ and are based on the Robot Operation System (ROS) [24]. Each operator and the area manager systems were run on separate machines. For the test vehicles, Freight mobile robots developed by Fetch Robotics, Inc. [25], were used. The robots have a nonholonomic constraint; the robots can only move forward or backward and turn in place and are unable to freely move in their lateral directions without

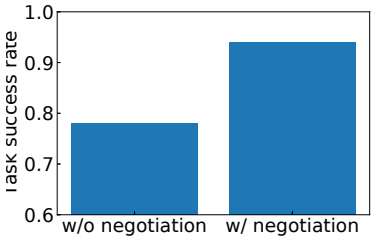


Fig. 8. Results of the experiments. The task success rate improved by introducing the proposed negotiation algorithm.

turning. Therefore, this constraint was incorporated during the path planning. The travel and rotation speeds of all vehicles were set to 0.2 m/s and 0.157 rad/s, respectively. These speeds correspond to the vehicles moving between two successive vertices and rotating $\pi/4$ radians, both within 5 s. The reservation duration of each path was conservatively set to 30 s for considering the uncertainties in the motion delays and localization errors of the physical robots.

The experiment was performed three times using the above settings. The task goals and deadlines were modified during each trial. The mean task success rate, task utility, and path cost over the three trials were computed for the case with and without negotiations.

In the experiments, 1.33 negotiations occurred during each task on average, and 33% of those negotiations reached an agreement. Fig. 8 shows the mean task success rate over all experiments. A task was treated as successfully completed when the assigned vehicle actually reached the goal by the deadline rather than the planned path did so . Using the proposed negotiation algorithm, operators could improve their success rate by approximately 20.5% on average. This result demonstrates the effectiveness of the proposed approach in a real environment and in online planning.

The systems mostly worked without any significant problems. Occasionally, however, the local planner was unable to generate a feasible path to the next waypoint when the robot detected other robots and treated them as obstacles in the cost map even after the actual robots moved away. When this occurred, it took a certain time to clear the cost map and find a feasible path, resulting in the robot being behind schedule. In addition, in the current implementation, each robot stopped its motion when a negotiation was requested. Although one negotiation finished in less than 1 s, it delayed the robot's planned path execution . Hence, we will update the system such that operators can plan and negotiate with each other without interrupting the robots' execution motion.

VI. Conclusions

In this paper, we addressed the path planning problem where multirobots with their own objectives are working in a shared space. We proposed a path negotiation algorithm to maximize an operator's task utility for such scenarios. The numerical simulations and physical experiments demonstrated the effectiveness of the proposed algorithm.

The proposed task-oriented utility function is quite general and can be applicable to purposes other than path negotiation addressed in this paper. The utility function can be, for instance, used in a path auction to evaluate bids from a task plan perspective.

Acknowledgment

The authors thank Yukiyasu Domae of the AIST for letting us use a Freight mobile robot and the research area for experiments. We also thank Dai Kinno of NEC for lending us their Freight robots for this study.

References

[1] Federal Aviation Administration, "Unmanned aircraft systems traffic management (UTM) concept of operations," 2018.
[2] Global UTM Association, "UAS traffic management architecture."
[3] C. Liu, et al., "Improving efficiency of autonomous vehicles by v2v communication," in 2018 ACC, pp. 4778—4783, 2018.
[4] M. Vasirani and S. Ossowski, "A market-inspired approach for intersection management in urban road traffic networks," J. Artif. Int. Res., vol. 43, no. 1, p. 621—659, 2012.
[5] O. Amir, G. Sharon, and R. Stern, "Multi-agent pathfinding as a combinatorial auction." in AAAI, 2015, pp. 2003-–2009.
[6] A. R. Pritchett and A. Genton, "Negotiated decentralized aircraft conflict resolution," IEEE Trans. on Intelligent Transportation Systems, vol. 19, no. 1, pp. 81-–91, 2018.
[7] J. Yu and S. M. LaValle, "Optimal multirobot path planning on graphs: Complete algorithms and effective heuristics," IEEE Trans. on Robotics, vol. 32, no. 5, pp. 1163—1177, 2016.
[8] R. Stern, et al., "Multi-agent pathfinding: Definitions, variants, and benchmarks," in SOCS 2019.
[9] D. Silver, "Cooperative pathfinding," in AIIDE, 2005, pp. 117-–122.
[10] G. Sharon, et al., "Conflict-based search for optimal multi-agent pathfinding," Artificial Intelligence, vol. 219, pp. 40-–66, 2015.
[11] O. Purwin, R. D'Andrea, and J.-W. Lee, "Theory and implementation of path planning by negotiation for decentralized agents," Robotics and Autonomous Systems, vol. 56, no. 5, pp. 422-–436, 2008.
[12] V. R. Desaraju and J. P. How, "Decentralized path planning for multiagent teams with complex constraints," Autonomous Robots, vol. 32, no. 4, pp. 385-–403, 2012.
[13] M. Čáp, et al., "Prioritized planning algorithms for trajectory coordination of multiple mobile robots," IEEE Trans. on Automation Science and Engineering, vol. 12, no. 3, pp. 835-–849, 2015.
[14] P. R. Wurman, R. D'Andrea, and M. Mountz, "Coordinating hundreds of cooperative, autonomous vehicles in warehouses," in IAAI 2007.
[15] H. Andreasson, et al., "Autonomous transport vehicles: Where we are and what is missing," IEEE Robotics Automation Magazine, vol. 22, no. 1, pp. 64—75, 2015.
[16] Z. Bnaya, et al., "Multi-agent path finding for self interested agents," in SOCS 2013.
[17] J. P. Wangermann and R. F. Stengel, "Optimization and coordination of multiagent systems using principled negotiation," J. of Guidance, Control, and Dynamics, vol. 22, no. 1, pp. 43-–50, 1999.
[18] D. Šišlák, et al., "Agentfly: Scalable, high-fidelity framework for simulation, planning and collision avoidance of multiple uavs," Sense and Avoid in UAS: Research and Applications, pp. 233—264, 2012.
[19] N. R. Jennings, et al., "Automated negotiation: prospects, methods and challenges," Int. J. of Group Decision and Negotiation, vol. 10, no. 2, pp. 199—215, 2001.
[20] K. Fujita, et al., Modern approaches to agent-based complex automated negotiation. Springer, 2017.
[21] Y. Mohammad, A. Greenwald, and S. Nakadai, "Negmas: a platform for situated negotiations," in Workshop on ACAN in IJCAI, 2019.
[22] R. Aydoğan, et al., "Alternating offers protocols for multilateral negotiation," in Modern Approaches to Agent-based Complex Automated Negotiation. Springer, 2017, pp. 153-–167.
[23] N. R. Sturtevant, "Benchmarks for grid-based pathfinding," IEEE Trans. on Computational Intelligence and AI in Games, vol. 4, no. 2, pp. 144-–148, 2012.
[24] M. QUIGLEY, "ROS : an open-source robot operating system," ICRA workshop on Open-Source Software, 2009.
[25] M. Wise, et al., "Fetch and freight: Standard platforms for service robot applications," in Workshop on Autonomous Mobile Service Robots, IJCAI, 2016.