Any-Com Multi-Robot Path-Planning: Maximizing Collaboration For Variable Bandwidth

Michael Otte and Nikolaus Correll

Abstract We identify a new class of algorithms for multi-robot problems called "Any-Com" and present the first algorithm belonging to that class "Any-Com intermediate solution sharing" (or Any-Com ISS) for multi-robot path planning. Any-Com algorithms find a suboptimal solution quickly and then refine that solution subject to communication constraints. This is analogous to the "Any-Time" framework, in which a suboptimal solution is found quickly, and refined as time permits. The current paper focuses on the task of finding a coordinated set of collision-free paths for all robots in a common area. The computational load of calculating a solution is distributed among all robots, such that the robotic team becomes a distributed computer. Any-Com ISS is probabilistically/resolution complete and a particular robot contributes to the global solution as much as communication reliability permits. Any-Com ISS is "Centralized" in the planning-algorithmic sense that all robots are viewed as pieces of a composite robot; however, there is no dedicated leader and all robots have the same priority. Previous centralized multi-robot navigation algorithms make assumptions about communication topology and bandwidth that are often invalid in the real world. Any-Com allows for collaborative problem solving with graceful performance declines as communication deteriorates. Results are validated experimentally with a team of 5 robots.

1 Introduction

Autonomous navigation is a key capability for enabling both industrial and consumer robotics to perform their work effectively. In fact, many of today's state-of-

Michael Otte
University of Colorado at Boulder, e-mail: michael.otte@colorado.edu

Nikolaus Correll
University of Colorado at Boulder, e-mail: nikolaus.correll@colorado.edu

the-art systems are being commercialized, and will become increasingly deployed into mainstream settings in the near future. As robot traffic becomes more congested, tomorrow's systems must be capable of coordinated interaction within a multi-robot society. This imposes a need for multi-robot navigation solutions that can plan efficient, coordinated, and collision-free paths for a collection of robots.

Complete solutions to multi-robot problems can be computationally complex. Although less expensive methods can enable practical performance in many real-world situations, these are incomplete and can fail in the most challenging circumstances (see Section 2.1). Often, each robot in a team is equipped with its own computer and the ability to communicate. Given these resources, it makes sense to divide computational effort among all robots a solution will benefit. That is, a networked team of robots can be re-cast as a distributed computer to solve the problems encountered by its composite robots. This is particularly useful for complex communal tasks such as centralized multi-robot path-planning.

In practice, wireless bandwidth is environment dependent and often beyond the control of the user or a system. Yet, algorithms for coordinating networked robot systems usually rely on a minimum quality of service and fail otherwise. We are therefore interested in distributed algorithms able to utilize unreliable communication, and coin the term "Any-Com" to describe them. The idea is to find a suboptimal solution quickly, and then refine toward optimality as communication permits. This is analogous to the "Any-Time" paradigm, in which algorithms adapt to the available computation *time* (Boddy and Dean, 1989). In this paper we present an algorithm called *Any-Com Intermediate Solution Sharing* (or Any-Com ISS) for performing centralized multi-robot path-planning within the Any-Com framework. In previous work, centralized solutions have either been calculated on a single robot and then disseminated, or solved by each robot individually (see Section 2.2).

In general, Any-Com algorithms exploit perfect communication and have gracefully performance declines otherwise. However, just as Any-Time algorithms cannot calculate a solution in 0 time, Any-Com ISS may not find a solution when communication totally fails. Worst-case scenarios aside, Any-Com ISS is robust to a high degree of communication disruption.

A brief survey of related work is presented in Section 2. Algorithmic details are provided in Section 3. In Section 4 we conduct a series of experiments both in simulation and on real robots. In Section 5 we discuss our results, and conclusions are given in Section 6.

2 Related Work

Here we briefly discuss a few multi-robot algorithms located along the communication, computation, and completeness spectrums. Recall that a *complete* algorithm is guaranteed to find a solution when one exists and will also report failure in finite time if a solution does not exist. A *resolution complete* algorithm is an algorithm that is complete to within a predefined granularity of the world representation. A

probabilistically complete algorithm is an algorithm that will find a solution, if one exists, in finite time with probability approaching 1.

2.1 Incomplete methods

In the *cocktail party* model each agent maintain its own world-view, goals, and navigation function, while remaining ignorant of other robots and their intentions (Lumelsky and Harinarayan, 1997; van den Berg et al, 2009). Each agent alternates sensing, planning, and movement, and there is no direct coordination between robots. While this algorithm is incomplete, it is popular due to simplicity, scalability, and minimal communication requirements.

In *prioritized planning* each robot's path is calculated separately, subject to the movement constraints imposed by the paths of higher-priority robots (Clark and Rock, 2001; Erdmann and Lozano-Perez, 1987; Hada and Takasa, 2001; Warren, 1990). Higher priority robots follow optimal to near-optimal trajectories while lower priority robots may be unable to find a solution. Prioritized planning has also been used to periodically create a line-of-sight communication chain while performing the somewhat related coverage task (Hollinger and Singh, 2010).

Decoupled planning breaks planning into two phases. In phase-1 each robot calculates its own path to the goal. In phase-2 the space-time positions of the robots along these paths are calculated such that no collisions occur (Aronov et al, 1998; Guo and Parker, 2002; Kant and Zuker, 1986; Leroy et al, 1999). Although decoupled planning can be distance-optimal, it is incomplete because each robot's path is completely determined after phase-1 (and they may pathologically conflict) (Sanchez and Latombe, 2002).

2.2 Complete methods: Centralized planning

In *Centralized planning* all robots are considered individual pieces of a single composite robot. Solutions are calculated in the resulting high dimensional configuration space. Robot paths are found by projecting the high-dimensional solution down into the relevant subspace per each robot. (Bonert, 1999; Clark et al, 2003; Sanchez and Latombe, 2002; Schwartz and Sharir, 1985; Xidias and Aspragathos, 2008). Previously, the high-dimensional solution has either been calculated by a single agent or at the same time on each robot (thus robots must communicate with this agent or each other, respectively). Centralized planning is theoretically complete but practical algorithms are usually probabilistically or resolution complete; nonetheless, it provides the best completeness guarantees of any multi-robot planning method.

2.3 Relevance to our work

Our Any-Com ISS algorithm (presented in Section 3) is centralized, and therefore shares many similarities to the work described above. One major difference is that previous work has not considered what happens when communication deteriorates—this is a main contribution of our work. Another important difference is that our algorithm leverages the distributed-computing power of the robotic team to help find better solutions more quickly. In contrast, previous work has required *each* agent to calculate an entire solution completely on its own.

Distributed versions of both prioritized planning and decoupled planning exist. For instance, in prioritized planning each robot can calculate its own path (assuming it respects robots of higher priority), and in decoupled planning each robot can individually calculate its own phase-1 solution (although these must be assembled by a single agent in phase-2). However, both prioritized planning and decoupled planning are incomplete, while Any-Com ISS is probabilistically/resolution complete.

We believe Any-Com ISS is most applicable to the complicated planning situations in which the incomplete planning methods fail, and advocate using the (less computationally complex) incomplete ideas under most circumstances. For this reason we only compare Any-Com ISS to state-of-the-art *centralized* planning techniques in Section 4—as these are the only other algorithms available when incomplete methods fail.

3 Methodology

Let the robot workspace $\mathbf{W}$ exist in $\mathbb{R}^2$. To guarantee probabilistic/resolution completeness, the entire team is considered a single composite robot. Each individual robot contributes 2 dimensions to the combined configuration space $\mathbf{C}$, in the form of position (x,y), and search occurs in a $\mathbb{R}^{2n}$ configuration space where n is the number of robots. We assume resolution accuracy δ is defined for the configuration state vector. δ is the minimum distance allowable between any two configurations per dimension and thus defines the resolution of the search. In a pragmatic sense, δ keeps the search-tree from being populated with essentially duplicate configurations, and focuses effort on finding (significantly) better solutions. We assume circular robots that can pivot in place, but note our algorithms can be generalized to arbitrary robots.

We use a heavily modified version of an any-time rapidly expanding random tree (RRT) inspired by Ferguson and Stentz (2006). Our underlying RRT differs from previous work (LaValle and Keffner, 2001) in two significant ways. First, instead of connecting a new node to the tree using the shortest possible edge, we use the edge that gives the new node the shortest possible distance-to-root. Second, instead of restarting each subsequent tree from scratch (i.e. while time remains to find a better solution), we prune the existing tree such that it only contains nodes that can possibly lead to better solutions—then continue growing the same tree subject to the constraint that new nodes must be able to lead to better solutions.

In general, we seek to utilize the distributed computational power of a team of mobile robots. We want algorithms that function in environments where communication is unreliable, but take advantage of reliable communication when it exists. To these ends, each agent maintains its own randomly created tree. Assuming *n* robots, the union of all trees is a $O(n)$ times larger tree maintained collectively by the entire team. Any-com is achieved by having robots share their individual intermediate solutions *during* path-planning so that all agents can prune globally sub-optimal branches from their local trees. This enables each robot to focus effort on finding only better solutions than those currently known to *any* robot. It also gives all robots a chance to directly refine the best intermediate solution. We call this idea Any-Com *Intermediate Solution Sharing* (Any-Com ISS).

Theoretically, allowing more agents to work on a random-tree problem will increase the chances a good solution is found quickly, regardless of whether or not the sharing of intermediate solutions has any affect. Therefore, to determine how much (if any) advantage Any-Com ISS provides, we compare Intermediate Solution Sharing to having *each* agent individually find a unique solution to the complete problem, then distributing them so the team can use the best one. We refer to the latter method as *Voting*, and note that similar ideas have been explored in the past (Clark et al, 2003). Finally, to give context to the relative performance of Any-Com ISS vs. Voting, we compare both of them to a client-server framework. In the client-server system, which we call *Baseline*, the server is charged with calculating a complete solution using a single random tree, and then sharing it with the other robots.

Any-Com ISS, Voting, and Baseline all use the same underlying random tree algorithm, shown in Figure 1-Left. To demonstrate that our random-tree algorithm performs well vs. previous work, we additionally compare results to Any-Time RRT (Ferguson and Stentz (2006)). We assume the existence of an admissible heuristic function $h(p_1, p_2)$ that returns the distance between configurations p_1 and p_2 ignoring any collisions. The value *bstln* stores the length of the shortest path known at any particular time. On line 4 we pick a new configuration p_1 to add to the tree—chosen as the goal with probability ρ and uniformly at random otherwise. On line 5, we check both if p_1 exists in $\mathbf{C}_{free}$, the collision free portion of the configuration space, and also if using p_1 can possibly lead to a better solution based on the start and goal configurations and *bstln*. Note that $\mathbf{C}_{free}$ is calculated with respect to both robot-robot collisions and robot-obstacle collisions. On line 7 we find the best node p_2 in the tree to use as a parent of p_1. We record $S_{dist}(p_1)$, the actual distance-to-start of p_1 through p_2, and then add p_1 to the tree on lines 10 and 11. If p_1 is the goal (and $p_2 \neq$ **null** on line 8) then the new path-to-goal is the best intermediate solution found so far, so we update *bstln* on line 13. On line 14 we use the function **findShortcuts**() to see if other nodes in the tree can reach the start more quickly via p_1 instead of their current parent. If so, we change the tree to reflect this, and update S_{dist} values of the descendants accordingly.

Lines 16-20 are only executed when Any-Com ISS is used. On line 16 we check for incoming messages from other agents that may contain better paths. If a better path is received, then it is added to the search-tree and *bstln* is updated (lines 18 and 19). Finally, we send messages to other agents on line 20.

RandomTree()

1: $bstln = \infty$
2: add *start* as root of search-tree
3: **while** $time < \mu$ **do**
4: pick a point $p_1 \in \mathbf{C}$, where $p_1 = goal$ with probability ρ
5: **if** $p_1 \notin \mathbf{C}_{free}$ **or** $h(start, p_1) + h(p_1, goal) \geq bstln$ **then**
6: **continue**
7: $p_2 =$ **findBestAndPruneTree**(p_1)
8: **if** p_2 = null **then**
9: **continue**
10: $S_{dist}(p_1) = S_{dist}(p_2) + h(p_1, p_2)$
11: add p_1 to search-tree as a child of p_2
12: **if** $p_1 = goal$ **then**
13: $bstln = S_{dist}(p_1)$
14: **FindShortcuts**(p_1)
15: **if** using Any-Com ISS **then**
16: check for messages at rate ω
17: **if** received better path **then**
18: add that path to search-tree
19: update *bstln*
20: send message with best-path

$p_2 =$ **findBestAndPruneTree**(p_1)

1: p_2 = null
2: $g_{p2} = bstln$
3: **for each** node $p_i \in$ Tree **do**
4: **if** $S_{dist}(p_i) + h(p_i, goal) > bstln$ **then**
5: remove p_i
6: **else if** p_1 is within δ of p_2 **then**
7: **return** null
8: **if** $S_{dist}(P_i) + h(p_i, p_1) < g_{p2}$ **then**
9: **if** edge $(p_i, p_1) \in \mathbf{C}_{free}$ **then**
10: $p_2 = p_i$
11: $g_{p2} = S_{dist}(P_i) + h(p_i, p_1)$
12: **return** p_2

FindShortcuts(p_1)

1: **for each** node $p_i \in$ Tree **do**
2: **if** $S_{dist}(p_i) + h(p_i, p_1) < S_{dist}(p_1)$ **and** edge $(p_i, p_1) \in \mathbf{C}_{free}$ **then**
3: $S_{dist}(p_1) = S_{dist}(p_i) + h(p_i, p_1)$
4: reroute p_i through p_1
5: **for** descendants of p_1 **do**
6: update $S_{dist}()$

Fig. 1: Random tree algorithm with flags indicating functionality native to Any-Com ISS (Left). Subroutine for finding p_2 (the best neighbor of p_1 already in the tree) and pruning the tree (Top-Right). Subroutine for checking if old nodes would do better by using p_2 as their parent (Bottom-Right).

While searching for p_1 in **findBestAndPruneTree**() (Figure 1-Right-Top) we simultaneously prune any nodes that cannot possibly lead to solutions shorter than *bstln*, and also check if p_1 is more than δ away from configurations already in the tree. Keeping the tree as small as possible focuses effort on finding better solutions.

Search continues until time μ, after which the most recent (and therefore best) intermediate solution is recorded as an agent's final solution. In Baseline, this is when the server distributes its final solution to the client robots, and also when individual solutions are compared in Voting.

We hypothesize Intermediate Solution Sharing will produce better solutions than the other two methods because it allows the entire team to have tighter search-tree pruning—focusing search toward new and improved solutions. Additionally, Any-Com ISS gives each agent the opportunity to improve the best solution found so far. Any-Com ISS is robust to packet loss because dropped messages do not affect an agent's ability to eventually find a solution. On the other hand, successful communication focuses search in beneficial ways and helps the team find better solutions

more quickly. Even out-of-date messages have the potential to be beneficial, as long as the solution they contain is better than the receiving agent's current best.

Each search-tree is generated randomly and each solution is drawn from a distribution over all possible solutions. Theoretically, both Any-Com ISS and Voting should increase the team's collective chances of finding a desirable solution, vs. Baseline, because n random samples are drawn from this distribution instead of 1.

Both Any-Com ISS and Voting use the same underlying message-passing protocol to disseminate information within the group. The idea is simple: each robot broadcasts information to every other robot at a predefined rate ω using the User Datagram Protocol (UDP). UDP drops unsuccessful messages, which keeps the information flowing through the network up-to-date. Each message contains the following information about the state of the global solution, based on the sending robot's current knowledge:

- Best solution (currently known to the sender)
- Best solution's length
- ID of the robot that generated the best solution
- List of robots that have submitted a final solution
- Movement flag
- List of robots that support best solution.

Each robot keeps a copy of what it believes to be the best solution found by any robot. Each robot is responsible for adding itself to the appropriate lists. In order to keep the network up-to-date, messages are dropped if they contain paths that are worse than the best path known to the receiving agent. Planning halts after time μ, at which point robots begin adding themselves to the list of robots that have submitted a final solution. Any robot can correctly deduce which agreement has occurred if it knows all robots have submitted a final solution (regardless of algorithm). This is because better solutions are no longer being generated and the best solution known to the sending robot is always sent in every message—the actual best solution must have been passed along with the knowledge that the robot who generated it has submitted a final solution. In the unlikely event of a tie, the solution found by the robot with the lower ID is used. Once a robot knows an agreement has been reached, it sets the moving flag to TRUE, begins moving along its path per the best solution, and rebroadcasts the best solution at ω. If a robot receives a message with a TRUE movement flag, it also starts moving and rebroadcasts that solution at ω.

Baseline modifies the method described above by setting the movement flag to TRUE as soon as time μ occurs. Therefore, each robot begins moving as soon as the solution is received from the server. In order to keep Baseline as naive as possible, the client robots do not rebroadcast the solution to each other, but the server continues to rebroadcast at ω.

Any-Com ISS also has an additional method of reaching an agreement. By carefully tracking which partial solutions the other robots most recently support (during the planning phase), it is possible to approximately forecast the final vote at time μ. After time μ, if a particular robot believes all robots currently support its most recent solution, then it starts moving on that solution and rebroadcasts it at ω (along with

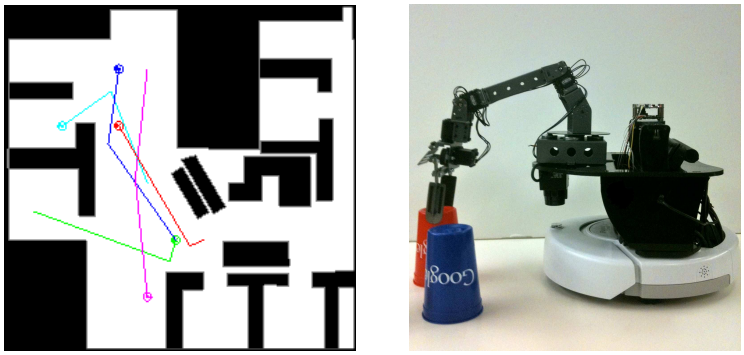

Fig. 2: A solution from Experiment 1 (Left). The Prairiedog Robotic Platform (Right).

the moving flag set to TRUE). This information is propagated through the network as usual (with disagreements broken toward solutions from robots with lower IDs). Although this protocol may allow a suboptimal solution to be chosen, it is unlikely. Further, if an agent erroneously believes *all* robots currently support its solution, then it must have had the best solution in the past, so the cost of erroneously picking a suboptimal solution is mitigated. A scenario where different robots move along different *incompatible* solutions is impossible because two or more robots cannot simultaneously believe all robots support their most recent solution. This is due to the fact that *only* the robot that generated a solution can initiate movement along it. If two different robots generate competing solutions, neither will initiate movement until one robot advertises support for the other's solution—and they cannot both support the other's solution because one solution is guaranteed to be better than the other (or, in the case of ties, come from the robot with lower ID).

4 Experiments

We perform two experiments with 5 robots in an office environment. Experiment 1 is conducted in simulation to evaluate theoretical performance over a wide range of parameters. Experiment 2 uses real robots to validate that the algorithms function in practice. Our robotic platform is the iRobot create, and we use the ROS operating system by Willow Garage. Robots are equipped with the Stargazer Indoor Localization System. Our Computational Units are System 76 Netbooks with built-in wireless networking capabilities.

Experiment 1 evaluates the relative performance of Any-Com ISS, Voting, Baseline, and Any-Time RRT (Figure 2). Note that Any-Time RRT is run on a single robot. We evaluate performance of all four algorithms vs. message success probability τ vs. planning time μ. We use $\tau = \{1, 1/4, 1/16, 1/64\}$ probability of success

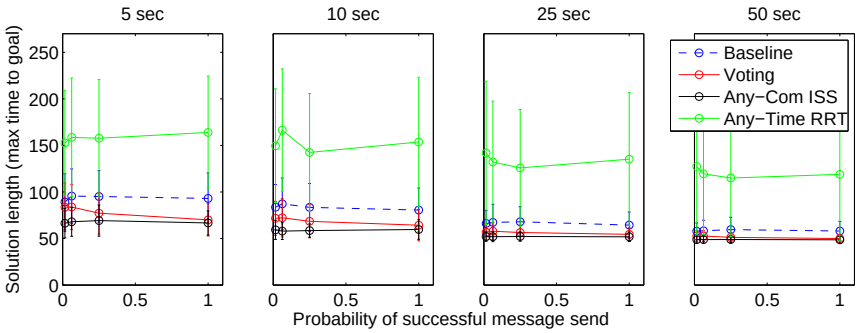

Fig. 3: Average Solution Lengths from Experiment 1. Sub-plots show different planning times μ.

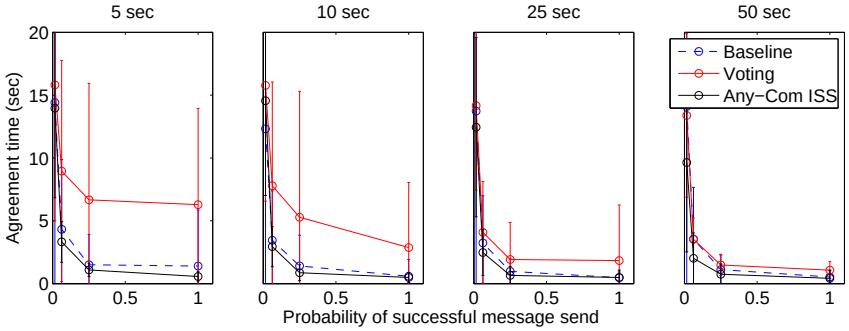

Fig. 4: Average agreement time from Experiment 1. Sub-plots show different planning times μ.

and $\mu = \{5, 10, 25, 50\}$ seconds. We perform 100 runs per each combination of parameters to facilitate statistical analysis of results. Mean and standard deviations of the resulting solution lengths are displayed in Figure 3 and agreement times in Figure 4 (agreement time is the time after μ and before movement). Note that agreement times are not presented for Any-Time RRT, since no message passing is required.

Experiment 2 is conducted on 5 actual robots and is similar to Experiment 1. Robot speed is 0.2 meters per second. During planning $\omega = 4$, and during the agreement phase $\omega = 32$. The change is due to the preliminary results in Experiment 1, where it is clear that the agreement phase can become lengthy in terms of messages sent. Also, path-planning is computationally intensive while the agreement phase is not, and robots are able to spare additional resources to increase ω. The same μ are used as in Experiment 1. Each data-point represents 20 runs. We plot solution quality and agreement time vs. planning time in Figure 5 Left and Right, respectively. Signal quality was relatively good in this experiment, the observed packet loss rate was less than 50%. We forgo comparison vs. Any-Time RRT due to the positive performance of the other methods in Experiment 1.

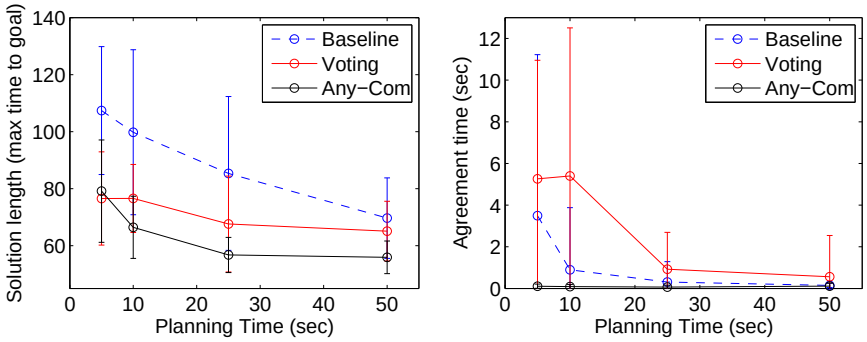

Fig. 5: Average Solution Lengths (Left) and average agreement time (Right) from Experiment 2.

5 Discussion of Results

With regard to solution quality, both Any-Com ISS and Voting out-perform Baseline, and Any-Com ISS outperforms Voting. All three methods outperform Any-Time RRT. Using a two-sample Kolmogorov-Smirnov test, we compare algorithms based on solution lengths, and find statistically significant ($p < .05$) differences between any two algorithms for all but one method-parameter combinations in Experiment 1 (i.e. for one method vs. another with μ and τ held constant), and all but one parameter combination in Experiment 2 (Voting vs. Any-Com ISS at $\mu = 5$ sec). In fact, $p < 0.001$ for most data-points in either experiment. When the results from all experiments are considered together, p becomes vanishingly small. These results validate our original hypothesis.

Examining the solution quality vs. planning time for the various methods in Figures 3 and 5-Left illustrate just how well Any-Com ISS performs. Voting finds similar quality solutions using less than half the planning time as Baseline, on average, while Any-Com ISS finds similar quality solutions in $\leq 1/n$ of the time! This is strong evidence the robotic team is functioning as an effective distributed computer. Given we are using n times as much computational power, the expected ratio of required planning time is $1/n$. Therefore, the super-efficient observed value of $< 1/n$ in Experiment 2 is impressive, especially given the minimal data shared between agents. Whether or not this trend will continue for larger groups of robots is a question we hope to answer in future work.

It often takes longer than 5 seconds (or even 10) for an agent to find a solution. That is, $\mu = 5$ is not enough time to guarantee that all robots have found a solution. In such a case, after 5 seconds has passed, Any-Com ISS uses the best solution found by any robot so far, while Baseline must wait until the server finds its first solution, and Voting must wait until all robots have found a solution. This has two interesting affects. First, the agreement times of Baseline and Voting are greater than Any-Com ISS because all robots must wait until the server or all robots have found a solution,

respectively, before an agreement can be reached. Second, by waiting extra time until n solutions exits, Voting has an increased chance of finding a "good" solution vs. Any-Com ISS. While this may initially seem desirable, we note that Any-Com ISS is able to start movement at the expected time, while the other algorithms suffer unexpected delays. We believe this is why the results for Voting and Any-Com ISS are similar for $\mu = 5$ sec in Experiment 2 (i.e. $p > .05$), and also why the agreement times for Voting and Baseline are inflated for $\mu = 5$ and $\mu = 10$ in Experiment 1.

Another interesting trend is that Any-Com ISS solution quality does not get much worse as communication becomes unreliable. Theoretically, as $\tau \to 0$ the results of Any-Com ISS will approach those of Voting. There is a hint of this in Experiment 1, where τ is controlled, especially for longer planning times. However, it appears communication must drastically deteriorate before Any-Com ISS begins to suffer. In fact, packet loss rates as high as 98% have little affect on solution quality.

The most noticeable effect of poor communication is an increase in the time it takes the robots to agree on a single solution. Assuming that communication failure is strictly Poisson-distributed, increasing the messaging rate ω during the agreement phase can mitigate the effects of communication deterioration (as we did in Experiment 2). In any case, the bandwidth will eventually become saturated, and further diminishing τ will eventually prevent an agreement from taking place within a useful time. Therefore, Any-Com ISS should not be used when $\tau \approx 0$. That said, it is impossible for *any* complete algorithm to function when $\tau \approx 0$. As a practical measure, the $\tau \approx 0$ case could be handled using a time-out. After which, robots start moving based on the best solutions known to them individually. Assuming on-board sensors exist, conflicts could then be resolved using the cocktail-party model. Although this 'worst-case-scenario' forces the algorithm to become incomplete until communication is resumed, it is arguably better than letting the team remain motionless forever. Further discussion on this idea is beyond the scope of this paper.

The simulated experiments predict Baseline should have similar agreement times to Any-Com ISS, while the real experiments show Any-Com ISS as the clear winner. The fact that these benefits do not extend to the Voting method (even for $\mu > 10$) suggests some other mechanism is responsible for the relatively quick agreement time of Any-Com ISS. We credit this improvement to the auxiliary vote-forecasting agreement method available to Any-Com ISS.

6 Conclusions

We coin the term "Any-Com" to describe algorithms that use multiple agents to collaboratively refine a solution toward optimality as communication permits. The motivation behind the general Any-Com idea is that distributed robots should adapt to use as much collaborative problem solving as communication quality permits. This is useful for solving computationally intensive problems, and especially well suited to problems with solutions of value to multiple agents. The problem domain of centralized multi-robot rover navigation has both of these qualities.

We present a practical Any-Com multi-robot path-planning algorithm called *Any-Com Intermediate Solution Sharing* (Any-Com ISS) in which agents share intermediate solutions so that the entire team can focus remaining effort on finding even better solutions. This works because it allows all robots to prune globally sub-optimal branches from their local search trees based on the best solution known to any member of the team. It also gives each robot an opportunity to directly improve the best solution. Intermediate Solution Sharing is Any-Com because dropped messages do not prohibit a solution from eventually being found, while successful messages improve solution quality (both in overall path quality, and the time it takes to reach an agreement). We envision Any-Com ISS as one tool among many in the multi-robot planning arsenal—useful in the specific case when a complete algorithm must be used (i.e. when a group of robots finds itself confronted with a difficult problem that cannot be solved by less expensive incomplete planning methods).

We perform 2 experiments using a team of $n = 5$ robots, and compare results to a basic server-client model as well as a voting method (in the server-client framework one agent plans and then distributes the solution to the other robots, while in voting each agent is allowed to plan separately and then the team uses the best solution found by any single agent). We find Any-Com ISS requires *less then* $1/n$ of the time required by the client-server framework to find a solution of similar quality, and less than $1/2$ the time required by the voting method, on average.

As bandwidth approaches 0 the solution quality of Any-Com ISS theoretically declines gracefully to that of the voting method, while both remain better than the server-client model. In fact, we find that communication loss as high as 98% has little affect on solution quality. Unfortunately, the time it takes to reach consensus approaches infinity as communication approach 0. This is not unexpected, as all complete algorithms are inherently vulnerable to *total* communication failure. Ignoring this worst-case-scenario, we find that Any-Com ISS is robust to a high degree of communication interference.

While this paper is a focused case-study on Any-Com applied to multi-robot navigation, we stress that the Any-Com idea is not limited to this particular domain. In particular, Any-Com ISS is applicable to any random-tree search through a metric space. We hope that the Any-Com concept will spread to other problems, and envision a world in which mobile robots dynamically take advantage all available computational resources to solve complex problems.

References

Aronov B, de Berg M, van der Stappen AF, Svestka P, Vleugels J (1998) Motion planning for multiple robots. In: Proceedings of the fourteenth annual symposium on Computational geometry, Minneapolis, USA, pp 374–382

Boddy M, Dean TL (1989) Solving time-dependent planning problems. In: Proc. Eleventh International Joint Conference on Artificial Intelligence, pp 979–984

Bonert M (1999) Motion planning for multi-robot assembly systems. MS dissertation, University of Toronto

Clark CM, Rock S (2001) Randomized motion planning for groups of nonholonomic robots. In: Proc. International Symposium of Artificial Intelligence, Robotics and Automation in Space

Clark CM, Rock SM, Latombe JC (2003) Motion planning for multiple mobile robots using dynamic networks. In: Proc. IEEE International Conference on Robotics and Automation, pp 4222–4227

Erdmann M, Lozano-Perez T (1987) On multiple moving objects. Algorithmica pp 477–521

Ferguson D, Stentz A (2006) Anytime rrts. In: Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems, pp 5369–5375

Guo Y, Parker LD (2002) A distributed and optimal motion planning approach for multiple mobile robots. In: Proc. IEEE International Conference on Robotics and Automation, pp 2612–2619

Hada Y, Takasa K (2001) Multiple mobile robot navigation using the indoor global positioning system (igps). In: Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, Hawaii, United States, pp 1005–1010

Hollinger G, Singh S (2010) Multi-robot coordination with periodic connectivity. In: Proc. IEEE International Conference on Robotics and Automation

Kant K, Zuker SW (1986) Toward efficient trajectory planning: the path -velocity decomposition. The international journal of robotics research 5:72–89

LaValle S, Keffner J (2001) Rapidly-exploring random trees: Progress and prospects. In: Algorithmic and Computational Robotics: New Directions, pp 293–308

Leroy S, Laumond JP, Simeon T (1999) Multiple path coordination for mobile robots: a geometric algorithm. In: Proc. International Conference on Artificial Intelligence

Lumelsky VJ, Harinarayan KR (1997) Decentralized motion planning for multiple mobile robots: The cocktail party model. Autonomous Robots 4:121–135

Sanchez G, Latombe JC (2002) Using a prm planner to compare centralized and decoupled planning for multi robot systems. In: Proc. IEEE International Conference on Robotics and Automation

Schwartz JT, Sharir M (1985) On the piano mover's problem iii. coordinating the motion of several independent bodies: the special case of circular bodies amidst polygonal barriers. In: Proc. IEEE International Conference on Robotics and Automation, pp 514–522

van den Berg J, Guy SJ, Lin M, Dinesh Manocha (2009) Reciprocal n-body collision avoidance. In: Proc. International Symposium on Robotics Research

Warren CW (1990) Multiple robot path coordination using artificial potential fields. In: Proc. of IEEE International Conference on Robotics and Automation, Cincinnati, OH, pp 500–505

Xidias EK, Aspragathos NA (2008) Motion planning for multiple non-holonomic robots: a geometric approach. Robotica 26:525–536