The Any-Com Approach to Multi-Robot Coordination

Michael W. Otte and Nikolaus Correll

Abstract— **We propose a new class of algorithms for multi-robot problems called "Any-Com". With Any-Com, a suboptimal solution is found quickly and refined as communication permits (analogous to "Any-Time" where a suboptimal solution is refined as time permits). Any-Com can be used to mitigate the high cost of solving multi-robot problems by dividing effort among all robots the solution will benefit. This is useful when complete algorithms are desirable. Previous algorithms make assumptions about communication that are often invalid in the real world. Any-Com enables collaborative problem solving as communication permits, with graceful performance declines otherwise. This paper provides a "recipe" for Any-Com algorithms and shows results from a multi-robot path-planning and a multi-robot task allocation problem that exemplify the approach.**

I. INTRODUCTION

Complete solutions to multi-robot problems can be computationally complex—e.g. complete multi-robot path planning, coverage, and task allocation problems have greater than polynomial runtime with respect to the number of robots. Although less expensive methods can enable practical performance in many real-world situations, these are incomplete and often not desirable when computational and communication resources that would enable more competitive solutions are available.

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 the effort of computing a solution among all robots the solution will benefit. That is, a networked team of robots can be re-cast as a distributed computer to solve problems encountered by its composite robots. This is particularly useful for complex problems—e.g. complete solutions multi-robot coordination problems. In previous work, such solutions have either been calculated on a single robot and then disseminated, or solved by each robot individually.

Wireless quality of service changes drastically over time and space. Bandwidth is environment dependent and often beyond the control of the user or a system. Yet, algorithms for coordinating networked robot systems (not to mention distributed systems, in general) usually rely on a minimum quality of service and fail otherwise. We would like robot teams to operate in realistic environments, and are interested in distributed algorithms that utilize available communication, but have gracefully performance declines otherwise. We coin the term "Any-Com" to describe this type of algorithm. An Any-Com algorithm's bandwidth utilization is analogous to an "Any-Time" algorithm's utilization of time [1].

We believe the Any-Com idea is applicable to a wide range of problems and two case-studies are presented in sections IV and V, respectively. The general theory is discussed in III, and we begin with a limited discussion of related work in section II.

Michael W. Otte and Nikolaus Correll are with the Department of Computer Science, University of Colorado at Boulder, 430 UCB, Boulder, Colorado 80309-0430 michael.otte@colorado.edu

II. RELATED WORK

Here we briefly discuss a few multi-robot algorithms that are located at either end of the communication, computation, and completeness spectrums. Any-Com algorithms fill this gap as they will max out available resources whenever possible.

A. Incomplete methods

Often it is advantageous to have each agent maintain its own world-view, goals, and navigation function, while remaining ignorant of other robots and their intentions. In the navigation domain, such algorithms are called the *cocktail party* model [2], [3], and can generally be described as greedy algorithms. Each agent alternates sensing, planning, and movement, and there is no direct coordination between robots. Such algorithms are incomplete, but are popular due to simplicity, scalability, and minimal communication requirements.

B. Complete methods: Centralized planning

In *Centralized planning* all robots are considered to be individual pieces of a larger composite robot. Solutions are calculated in the resulting high dimensional composite configuration space (e.g. [4]–[9] for the multi-robot path-planning domain). Similarly, in [10] a market-based algorithm is used to allocate tasks in a multi-robot coverage problem. Although this algorithm is robust to communication failure — leading to all robots performing all tasks — it is not Any-Com as communication is not used to distribute computation. Indeed although being the most closely related works to our own, in all of the above algorithms each agent must calculate an entire solution on its own (or calculated on a single robot and then distributed to the others). In contrast, Any-Com algorithms should leverage the distributed-computing power of the robotic team to help find better solutions more quickly.

III. THE ANYCOM APPROACH

Any-Com algorithms assume robots have the ability to send and receive messages. However, they must function in realistic communication environments, so only non-critical

parts of an algorithm should be parallelized. This allows successful communications to help better solutions to be found more quickly, while unsuccessful communications do not hinder the eventual discovery of a solution. For example, if the algorithm involves searching a state space, the problem can be divided such that each robot is likely to search a unique part of that space. This works best if the problem being solved is likely to have many solutions—possibly of varying quality, instead of just one.

If the solution is used for real-world coordination, we must assume a small finite number $O(n)$ of communications are successful, where n is the number of robots. This is required to guarantee all robots execute the same solution. We note that total communication failure ($< O(n)$ successful messages) cannot be addressed by any complete multi-robot algorithm. The only option in these cases is to fall back on incomplete methods. Although we do not address total communication failure, Any-Com algorithms can be robust to significant communication deterioration (e.g. $> 95\%$ chance a packet is dropped).

Because the entire team collaborates in real-time, an attempt should be made to keep each member up-to-date with respect to the rest of the team. For instance, many algorithms can benefit from sharing intermediate data. Specifically, if the team is building a search tree, e.g. for multi-robot navigation, the tree can be pruned based on the length of the best solution currently known. Thus, sharing information about the current best allows the entire team to focus energy on finding better solutions. In practice we use a simple UDP based protocol so that new up-to-date messages are always sent and dropped messages are ignored.

Data from one robot may be useful to any/every other robot, and information is propagated throughout the team. This does not mean a particular message reaches every other robot, but rather, each robot updates its internal view of the current problem, and then broadcasts what it believes to be the critical information. Over time, useless information is ignored while important information is rebroadcast until it is outdated and replaced.

IV. MULTI-ROBOT PATH PLANNING

The multi-robot navigation problem is to find a coordinated set of collision-free paths for all robots moving within a common area. We discuss three algorithms: an Any-Com idea *Partial Solution Sharing*, a simple distributed framework *Solution Sharing* and a standard client-server framework used for comparison *Baseline*. All algorithms use an Any-Time rapidly-expanding random tree (RRT) [1], and all robots are allowed to plan for some minimum time μ before moving.

A. Methodology

In *Solution Sharing*, all robots search separately for a solution to the multi-robot path-planning problem. After μ, the best over-all solution is dispersed. *Partial Solution Sharing* extends this idea by sharing intermediate Any-Time solutions as they become available. Messaging begins once the first (sub-optimal) solution is known to the sending robot. In *Baseline*, a single server robot plans for time μ and then sends the solution to the other robots.

The same RRT is used as the base algorithm for Solution Sharing, Partial Solution Sharing, and Baseline. The algorithm works by randomly picking a point p_1 in the current search-tree and attempting to connect it to a new random point p_2 in the configuration space, subject to the constraints imposed by the robots' dynamics. With probability ρ the coordinates of p_2 due to a particular robot are chosen along the straight-line path to that robot's goal, otherwise they are chosen at random.

Once the first intermediate solution is found, we prune the tree by removing sub-paths that cannot possibly lead to better solutions. Aggressive pruning focuses the search and helps new (i.e. better) intermediate solutions to be found more quickly.

Search continues until time μ, when the most recent (and therefore best) intermediate solution is recorded as an agent's final solution. The solutions of all robots are distributed and the best is used. In Baseline, the server is the only robot with a solution to distribute.

In Partial Solution Sharing, agents share intermediate solutions with each other. This allows the entire team to have tighter search-tree pruning, and further focus search toward new and improved solutions. Additionally, it gives each agent the opportunity to improve the best solution found so far. Intermediate solution sharing is robust to communication failures because dropped messages do not affect an agent's ability to eventually find a solution. On the other hand, successful communications focuses the search in beneficial ways and help the team find better over-all 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.

Because the search-tree is generated randomly, each solution is drawn from a distribution over all possible solutions. Solution Sharing increases the team's collective chances of finding a desirable solution, vs. Baseline, because N random samples are drawn from this distribution instead of just one.

B. Experiments and Results

We perform two experiments. The first is conducted in a simulated environment and the second on a team of 6 real robots. Simulation is used for Experiment 1 in order to evaluate theoretical performance over a wide range of parameters. Real robots are used for experiment 2 to validate that the algorithms function in practice.

C. Experiment 1

Experiment 1 consists of six simulated robots operating in an obstacle free environment. The robots start equally spaced around a circle facing inward (Figure 1, Left). The goal is to have all robots end up on the opposite side, also facing inward (Figure 1, Right). This causes a high degree of congestion in the center of the environment. We evaluate the performance of all three algorithms vs. message success probability τ, vs. planning time μ. In order to facilitate

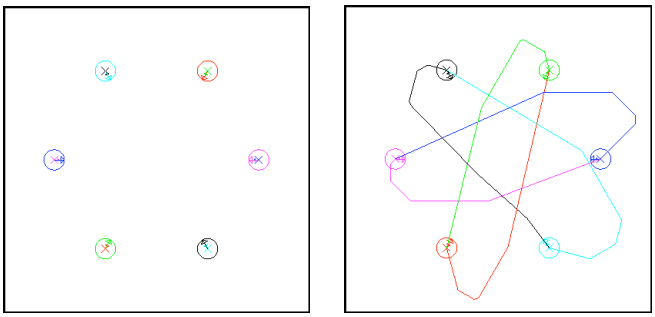

Fig. 1: Experiment 1, Circles with large arrows represent robots, 'x's with small arrows represent goals. Starting configuration (Left), and a resulting solution (Right).

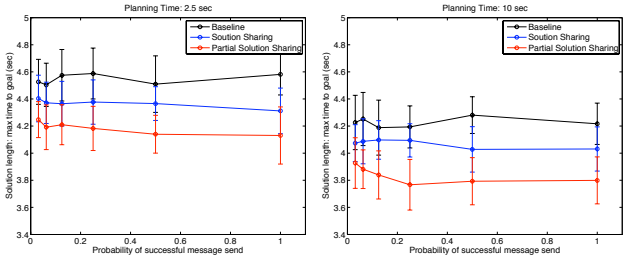

Fig. 2: Average Solution Lengths from Experiment 1.

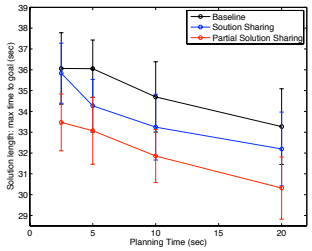

Fig. 3: Average Solution Lengths from Experiment 2.

statistical analysis of results, we perform 20 runs per each combination of parameters. Mean and standard deviations of the resulting solutions are displayed in Figure 2.

D. Experiment 2

Experiment 2 is conducted on 6 actual robots and is similar to Experiment 1. During planning we set ω to be 4. We perform tests using the same planning times μ as in the previous experiment, and perform 10 runs per experiment per point. We plot solution quality in Figure 3.

E. Results

Partial Solution Sharing outperforms the other two algorithms and Solution Sharing out-perform Baseline. 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 the vast majority of method-parameter combinations (i.e. for one methods vs. another with μ and τ held constant). Baseline and Partial Solution Sharing are always statistically different ($p < .003$). When all experiments are considered together, p becomes vanishingly small for Experiment 1 and less than .0002 for Experiment 2.

To clarify just how well the Any-Com Algorithms perform we note that, on average, Solution Sharing finds similar quality solutions using half the planning time as Baseline, while Partial Solution Sharing finds similar quality solutions *in one quarter of the time*! This is strong evidence that the robotic team is functioning as an effective distributed computer. Given that we are using $N = 6$ times as much computational power, the lower bound on the ratio of required planning time is $1/6$. However, our observed value of at most $1/4$ is impressive given the minimal amount of information shared between agents.

Another interesting trend is that solution quality does not get much worse when communication becomes unreliable. Theoretically, as $\tau \to 0$ the results of Partial Solution Sharing will approach those of Solution Sharing. There is a hint of this effect in our results, especially for longer planning times. However, it appears communication must drastically deteriorate before Partial Solution Sharing begins to suffer.

V. MULTI-ROBOT TASK ALLOCATION

The multi-robot task allocation problem is concerned with how to distribute r robots to l locations. In this paper we are concerned with the spacial case $r = l$, but note that our methods can easily be applied to cases where $r \neq l$. We also assume that all robots are identical with respect to their functionality vs. a particular goal. That is, any robot is equally qualified to be deployed to any goal. We seek a solution that minimizes the total time required for all goals to receive a robot. In the special case $r = l$ the number of possible robot vs. goal pairings reduces to $(r!)$. Each solution may or may not be possible, and some are more likely to be better than others.

A. Methodology

In general, each set of robot vs. goal pairings is itself a multi-robot path planning problem as the path length from robot to task is part of the task-cost. Therefore, we build Any-Com task allocation on top of Any-Com path planning. One can envision a number of way this could be done, especially since the problem space is so large. While we hope to explore many other ideas in the future, here we focus on one simple algorithm to illustrate the Any-Com design philosophy. Like in the path-planning example, each robot selects a different set of robot vs. goal locations and attempts to find the best multi-robot path planning solution given those constraints.

In order to improve the likelihood to evaluate "good" permutations early and to ensure that robots work on different areas of the search space, we use a heuristic to divide the task-allocation search space. Each robot sorts all permutations of start vs. goal location based on the minimum possible solution assuming straight start-goal paths for all robots (Euclidian distance). Next, the top r candidates are divided among the top r robots based on robot ID number. Alternatively, and analogue to the RRT algorithm used in the path-planning example, this could be achieved

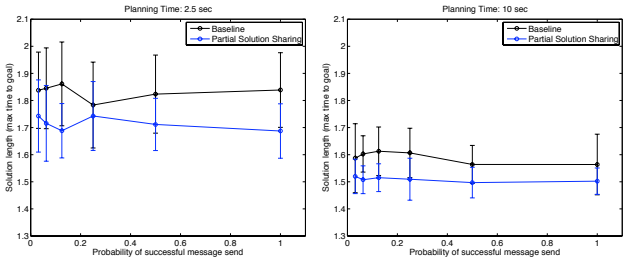

Fig. 4: Average Solution Lengths from Experiment 3.

by assigning a random permutation to each robot. Like in the RRT algorithm, this approach bears the, albeit small, risk that two robots evaluate the same permutation incidentally, however. We compare Any-Com results to a client server framework where one robot calculates a solution based on the best heuristic-based set of robot vs. goal locations. The latter system is called Baseline in this experiment, while the Any-Com solution is called partial solution sharing.

B. Experiments and Results

One experiments (Experiment 3) is performed in simulation as a proof of concept. There are 6 robots that must get to 6 locations. The robots start in two vertical lines on the edges of the workspace and must form a horizontal line in the center of the workspace. As with experiments 1 and 2, we evaluate performance given message success probability τ and planning time μ. We perform 20 runs per each combination of parameters and algorithm. Mean and standard deviations of the resulting solutions are displayed in Figure 4.

Experiment 3 shows that Any-Com provides significant improvement ($p < .001$) over the client server model in all cases. The Any-com algorithm finds similar quality solutions in roughly half the time than the client server model. As with Experiment 1, solution quality degrades gracefully as communication becomes unreliable.

VI. CONCLUSIONS

We propose the Any-Com paradigm and present two example Any-Com algorithms—one for multi-robot path-planning and one for multi-robot task allocation. The motivation behind 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 involving multiple robots. Whereas some multi-robot methods can function without communication, they must restrict themselves to using only local information. As such, they are unlikely to find globally optimal solutions and may even fail when valid solutions exist. Whereas complete algorithms are computationally expensive, they must be used in worst-case scenarios. In these situations, it makes sense to divide the computational effort among all robots the solution will benefit. We note that complete solutions to both the proposed NP-hard problem domains scale poorly with increasing team-size. While scalability classically is concerned only with computation, we argue that the true bottleneck in such a scenario will be the ability to communicate. Thus, an important aspect of Any-Com algorithms is their capability to fall back to solutions that do not require any communication. For path-planning this is the cocktail party model, and for task allocation, this is random allocation.

When designing Any-Com algorithms, non-critical parts of the algorithm should be parallelized so that communication disturbances do not prohibit an eventual solution from being found, and successful communication helps better solution to be found more quickly. Any-time algorithms are well suited for modification into Any-Com algorithms because they find a sub-optimal solution quickly then refine that solution as time permits. The Any-Com approach adds additional computational resources, along with a means of communication, allowing the initial search and subsequent refinement to happen in-parallel. This multiplies the amount of solution refinement realized per time.

Our case-studies show that Any-Com provides significant improvement vs. classical centralized methods. While this investigation was focused on multi-robot navigation and task-allocation, we stress that Any-Com idea is not limited to this particular domain. Ideally, any robot in a particular area should be able to donate computational resources to help solve whatever collaborative problems are beneficial to its mission. In future work we are interested in applying the Any-Com paradigm to other classes of multi-robot coordination problems as well as in developing a theoretical framework for its analysis.

REFERENCES

[1] D. Ferguson and A. Stentz, "Anytime rrts," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2006, pp. 5369–5375.

[2] V. J. Lumelsky and K. R. Harinarayan, "Decentralized motion planning for multiple mobile robots: The cocktail party model," *Autonomous Robots*, vol. 4, pp. 121–135, 1997.

[3] J. van den Berg, S. J. Guy, M. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," in *Proc. International Symposium on Robotics Research*, 2009.

[4] Robotica, "Motion planning for multiple non-holonomic robots: a geometric approach," *Robotica*, vol. 26, pp. 525–536, 2008.

[5] M. Bonert, "Motion planning for multi-robot assembly systems," *M.S. dissertation, University of Toronto*, 1999.

[6] J. T. Schwartz and M. Sharir, "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*, 1985, pp. 514–522.

[7] G. Ramanathan and V. S. Alagar, "Algorithmic motion planning in robotics: coordinated motion of several disks amidst polygonal obstacles," in *Proc. IEEE International Conference on Robotics and Automation*, 1985, pp. 514–522.

[8] C. M. Clark, S. M. Rock, and J.-C. Latombe, "Motion planning for multiple mobile robots using dynamic networks," in *Proc. IEEE International Conference on Robotics and Automation*, 2003, pp. 4222–4227.

[9] G. Sanchez and J.-C. Latombe, "Using a prm planner to compare centralized and decoupled planning for multi robot systems," in *Proc. IEEE International Conference on Robotics and Automation*, 2002.

[10] P. Amstutz, N. Correll, and A. Martinoli, "Adistributed boundary coverage with a team of networked miniature robots using a robust market-based algorithm," *Annals of Mathematics and Artifcial Intelligence*, vol. 52(2-4), pp. 307–333, 2009.