Consensus-Based Task Sequencing in Decentralized Multiple-Robot Systems using Local Communication

Chris A. C. Parker and Hong Zhang

Abstract—Behavior-based controllers for complex missions often are more easily designed by decomposing the mission into a series of smaller subtasks. When applying this technique to a multiple-robot system, the entire system should focus its work on one subtask at a time to prevent interference from robots working on conflicting subtasks simultaneously. We present a behavior-based approach to the task-sequencing problem for decentralized multiple-robot systems requiring only local inter-robot communication. Consensus is used to prevent premature task sequencing. Our algorithm is compared to globally-communicative and communication-free strategies. The proposed task-sequencing behavior is found to be efficient with respect to time and energy requirements, and could be applied to many existing decentralized multiple-robot systems.

I. INTRODUCTION

Dividing a complex mission into a sequence of simpler subtasks substantially simplifies the design of a behavior-based controller [3]. Instead of designing one complex controller that covers an entire task, a set of simpler controllers is built: one for each subtask. These would then run in order, control passing from one to the next as each subtask was completed. This approach promotes code reuse, as the simpler sub-controllers are more likely to be used again for future missions. Because behavior-based controllers become increasingly difficult to debug as they grow, this divide-and-conquer approach also would tend to produce more robust control in general.

This approach assumes that a system is able to detect the completion of each subtask and automatically move onto the next subtask in its queue. For solitary robots or centrally controlled systems, task sequencing is trivial. A solitary robot or central controller need only detect the completion of the current subtask and then initiate the switch to the next subtask. For decentralized systems [5], in which global behavior is emergent, this is not such a simple operation. Behavior-based control is common in decentralized multiple-robot systems (MRS), so a behavior-based strategy to step a decentralized MRS through a sequence of tasks is desirable. In this paper, we present such a task sequencing behavior for use in decentralized MRS and demonstrate its performance in a site preparation domain.

A. Task Sequencing

Consider a mission \( M \) composed of a sequence of subtasks \( m_i \) as shown by Equation 1.

\[
M = [m_1, m_2, \ldots, m_i, m_{i+1}, \ldots, m_n]
\]  

To complete \( M \), a system must complete each subtask in order. In some scenarios, work on some subtasks must be completed and halted before any work on the next can commence. Consider a painting mission composed of priming followed by painting. Were paint applied before priming were completed, the mission would fail. Furthermore, were some robots to continue priming once that subtask had been completed and others had begun to paint, the mission would similarly fail. We refer to such subtasks as mutually exclusive. A team carrying out a sequence of mutually exclusive subtasks must coordinate its transitions from one to the next.

Even when a sequence of subtasks is not mutually exclusive, it sometimes is advantageous to step a system from one to the next as a cohesive team, as this will tend to maximize the rate at which each subtask - and in turn the entire mission - is completed.

Without inter-robot communication, task-sequencing is difficult to coordinate. Each robot must independently choose when to move on to the next task. This will occur at a different time for each robot and they will tend to spread themselves across a mission’s subtasks. For this reason, non-communicative systems are unable to accommodate mutually exclusive subtasks.

Global communication (i.e. MRS whose robots can communicate with sufficient single-hop range to reach all of their teammates) greatly simplifies the task sequencing problem. Once a robot detects the completion of the current subtask, it broadcasts this to the rest of its team which synchronizes the entire system’s transition to the next subtask. As system population increases, so does the probability of a robot making an error and prematurely stimulating the switch to the next subtask. Consensus can counter this problem. Work on the next subtask could be delayed until a preset number of robots - a quorum - had each detected the completion of the current task and broadcast this to the rest of their team. Mutually exclusive subtasks easily can be accommodated through global communication.

Robots in a locally communicative system have single-hop communication ranges that are less than their system’s physical diameter, and thus cannot reach every robot with a single transmission. In some cases, the communication range is so short that robots must be right next to each other to exchange messages. Such systems are very disconnected, requiring their robots to move about to communicate with each other. This is typical of systems composed of very small robots, where physical movement becomes more energy efficient than long-range transmissions [4]. Flooding algorithms...
allow information to be shared information across such a system, but the problem of consensus still remains. Robots must measure the degree of consensus within their team, and only once a quorum of robots agree that the current subtask is complete should they induce their system abandon the current subtask in favor of the next one.

B. Site Preparation via Blind Bulldozing

Before a structure can be built, the building site must be cleared of debris to permit construction to commence. This clearing is known as site preparation [8]. In an earlier work, we described a multiple-robot site preparation algorithm called blind bulldozing [11] based on the cooperative nest construction behavior of the ant Temnothorax albipennis [6]. The individual robots in this algorithm autonomously bulldoze rock and debris from a construction site. Their controllers are simple: push debris in a straight line and reorient to a random heading when the robot either encounters a team mate or the force on their plow exceeds a threshold. Over time, a circular area is cleared by the randomly distributed plowing. The task is complete once the cleared area achieves a preset size. This can be inferred by the distance that a robot travels in between its reorientations. Secondary construction must not begin until all of the bulldozing has ceased or else the secondary structures would continually be toppled by any robots still bulldozing. Site preparation and secondary construction are a mutually exclusive task pair.

C. Related Work

The union of behavior-based control [3] and decentralized MRS [5] is a powerful one, as both are based around same basic concept: useful behavior can emerge out of the interaction of numerous lesser behaviors. Natural social systems like ant colonies also are organized around this concept, and they have served as the inspiration for numerous decentralized MRS [2], [7]. Some examples of social insect-inspired applications include collective transport [9] and collective construction [13]. Like these, the majority of decentralized MRS have been single-purpose entities. They are placed in an environment, activated, and the solution to the target problem emerges out of their interactions. It would be useful to treat a decentralized MRS as a single intelligent entity [10]. This necessitates group decision-making, and there has been some work done in this area. Wessnitzer and Melhuish presented a simulated robotic swarm that could select as a team which target to pursue [14], while Kok and Vlassis have demonstrated more complex group decision-making for play-selection in smaller MRS in the RoboCup domain. We also demonstrated decentralized group decision-making in a collective relocation domain based on the nest site selection behavior of Temnothorax ants [10]. The consensus-based task sequencing presented in this work aims to extend group decision-making in decentralized MRS in a different direction, allowing a team to make a cooperative decision about the current state of their mission and use this to coordinate their transition to the next task.

II. DECENTRALIZED TASK SEQUENCING WITH LOCAL COMMUNICATION

The use of consensus is central to our approach. Before a MRS or swarm abandons its current task, some predetermined proportion of its members should agree that the task is complete. This will reduce the probability of premature task sequencing as the effect of individual errors in task-state assessment will be moderated. Simultaneous work on consecutive subtasks should be prevented so that mutually exclusive subtasks can be accommodated.

![Task Sequencing Diagram](image_url)

Fig. 1. This is an overview of our task sequencing strategy. Robots begin in the working state, but enter the advocating state when they independently determine the current task to be complete. Once a robot in the advocating state determines that a sufficient proportion of its team is advocating (and thus agrees that the current task is complete), it commits and induces the remainder of its team to do so as well. When robots no longer can find uncommitted robots, they begin work on the next group task. The addition of the advocating and committed states allow a decentralized MRS with short range communication to coordinate its transition from one task to the next in a sequence.

We will refer to robots contributing to the current subtask as those in the working state and those that have moved onto the next subtask as being in the finished state. Our strategy inserts two additional states between these: advocating and committed. Refer to Figure 1. All of the robots begin in the working state and contribute to the current subtask. When a robot determines that the current subtask is complete, it enters the advocating state and halts its work on the current subtask, but it continues to wander amongst its teammates. When advocating robots encounter a teammate, they send it a vote-message. The greater the advocating population, the more often vote-messages will be received by individuals in the system. It is through the reception of these messages that the degree of consensus is estimated, which we will detail in Section II-A. Once an advocating robot believes that a sufficient consensus or quorum has been established, it enters the committed state. Committed robots also halt their work on the current subtask and wander throughout their system, but instead of vote-messages, they send commit-messages. Robots in either the working or advocating states that receive these immediately enter the committed state and acknowledge the commit-message with a response. Robots in the committed or finished states ignore them. Once a
robot has been in the committed state for a preset period of time without receiving a response to its commit-messages, it enters the finished state, at which point it begins work on the next subtask. The advocating and committed states serve to synchronize the decentralized system and insert a dead-band during which no work is done on either the current or next subtasks in order to accommodate mutually exclusive subtasks.

A. Consensus Estimation

In this section, we describe the strategy used by the advocating robots to measure the degree of consensus in their system. We focus entirely on short-range, anonymous unrouted messages.

![Regularly Incremented Decaying Variable: Two Rates of Increment](image)

Fig. 2. This figure illustrates the effect of regularly adding a constant to an exponentially decaying variable at two different rates. Both lines decay exponentially via the same time constant, and are incremented regularly by the same amount. The dashed line is incremented every 8 seconds, while the solid line is incremented only every 16 seconds. If the time constant and value of increment are known, the peak values of the two curves can be used to infer the rate at which they are incremented.

At first, simply counting the number of other vote messages received might seem satisfactory. However, for this method to be effective, each robot must be uniquely identifiable and this information would have to be included with each vote-message to prevent the robots from voting multiple times in each other’s polls. Robots that communicate using simple semaphores such as those described in [1] would be unable to employ the counting strategy.

Our proposed strategy takes its cue from social insect behavior. Some ants are able to measure their rate of encounter with teammates [12], and it is likely that they do so without explicit calculation. Rate of encounter is a measurement of population density. We utilize an exponentially decaying variable to convert encounter rate into an amplitude that can more easily be thresholded. Consider the function \( q(t) = e^{t/\tau} \). The rate at which it decays is proportional to its magnitude. If some constant \( \Delta \) were added to \( q(t) \) at a regular interval, a sawtooth pattern would be produced that would reach an equilibrium determined by the length of the interval, \( \Delta \) and \( \tau \). This principle is illustrated by Figure 2. To use this to test whether quorum has been met in the advocating population, \( \Delta \) is added to \( q(t) \) with the reception of each vote-message. Testing quorum is reduced to a thresholding operation: Once \( q(t) \) exceeds a threshold calibrated to a desired density of advocating robots, quorum is deemed satisfied and the robot enters the committed state\(^1\).

The first step is to determine \( \tau \). This is related to the period between a robot’s encounters with teammates, \( T_0 \), and Equation 2. If \( \tau \) is too small, \( q(t) \) will decay too rapidly and the different encounter rates will be indistinguishable. If it is too large, it will take a larger number of encounters (and thus more time) with advocating teammates in order to reach an appropriate threshold. In this work, we have found that the relationship in Equation 2 strikes an appropriate balance between these two concerns.

\[
\tau = -\frac{T_0}{\ln(1 - e^{-2})} = T_0/0.1454 \quad (2)
\]

In an \( n \)-robot system in which \( n_a \) robots are in the advocating state, the expected period of time between each vote-message received by an advocating robot is given by Equation 3.

\[
T_e = \frac{T_0(n - 1)}{n_a - 1} \quad (3)
\]

Finally, the maximum value reached by \( q(t) \) at equilibrium as a function of the advocating population is given by Equation 4.

\[
q_{max}(n_a) = \frac{\Delta}{1 - e^{-\frac{\Delta}{T_0}}} = \frac{\Delta}{1 - e^{-0.1454\left(\frac{1}{n_a}\right)}} \quad (4)
\]

\( q_{max}(n_a) \) is plotted in Figure 3 for a 15-robot system, which reveals a nearly linear relationship. Because \( q(t) \to q_{max}(n_a) \) as \( t \to \infty \), setting the threshold for \( q(t) = q_{max}(n) \) is not a good idea. Rather, the advocating robots should threshold on some fraction of \( q_{max}(n_a) \) small enough to ensure commitment in a reasonable amount of time, yet large enough to make premature commitment unlikely.

It is important to remember that decentralized MRS are stochastic entities: \( T_0 \) is a random variable, not a single-valued constant. We have found that using the first quartile of \( T_0 \) in the above equations produces satisfactory performance. Nonetheless, we should expect and be able to accommodate some error in the consensus estimation process. It is possible to choose a threshold so high that no possible degree of consensus will lead to commitment, so care should be exercised during its selection.

III. EXPERIMENTS

Our proposed task sequencing algorithm was implemented in simulation in order to analyze its behavior. Two additional approaches were implemented for the purpose of comparison. We will refer to our proposed behavior as the locally

\(^1\)In a micro- or nano-robot, this could be implemented with an RC-circuit. Applying a constant current source to the capacitor for a fixed period of time would increase its voltage by a constant which would then decay exponentially.
communicative approach. The first of the other two implementations was a non-communicative approach in which the robots independently transition from the working to finished state. The second employed global communication. The robots independently enter the advocating state when they believe the current task complete, at which point they globally broadcast a vote-message to the rest of their team. Once a predetermined number of vote-messages have been received the robots all enter the finished state.

The locally communicative system was calibrated as follows. First, a simulation of randomly wandering robots was run and \( T_0 \) was set to the first quartile of the inter-robot encounter period: 3.8 seconds. \( \tau \) and \( q_{\text{max}}(n) \) were computed using \( T_0 \) according to Equations 2-4. The actual thresholds used for quorum testing were computed according to Equation 5.

\[
q_{\text{max}}(n_a) = 0.95(q_{\text{max}}(n_a) - q_{\text{max}}(n_a - 1)) + q_{\text{max}}(n_a - 1)
\]

(5)

The robots’ communication range was set to 0.65 meters, meaning that two robots could communicate only if their outer hulls were within 0.15 meters of each other. Once a robot had been in the committed state for 180 seconds without receiving a response to its commit-messages, it would conclude that there were no more uncommitted robots and would enter the finished state. The globally and locally communicative robots were programmed to commit at quorums of 1, 6, 9 and 12 advocating robots. Each system configuration was run 25 times, and each trial was ended after 2500 seconds regardless of the progress achieved.

All of the systems were implemented in a blind bulldozing environment using the TeamBots simulation package. The environment consisted of a circular arena 12 meters in diameter inside of which the robots wandered about along straight lines, reorienting to random headings when and obstacle was encountered. Each system was composed of 15 circular robots of 0.5 meters in diameter. The robots determined that the blind bulldozing was complete after they had traveled 11 meters without encountering an obstacle.

IV. RESULTS AND ANALYSIS

The locally and globally communicative systems both completed the task sequencing behavior within 2500 second time limit in 100% of the trials. Only 6 out of the 25 non-communicative trials completed before the 2500 second time limit, implying that on average, the non-communicative robots would have required more than 2500 seconds to complete the transition to the next task.

The qualitative behavior of the locally communicative task sequencing scheme can be seen in Figure 4. One at a time, the robots enter the advocating state as they are able to travel 11 meter paths without reorienting. As more robots advocate, it can be seen that the advocating robots begin to receive vote-messages, and their rate of reception increases with advocating population. Eventually, an advocating robot detects quorum and commits, and the other robots quickly follow suit as commit-messages flood the system. Once all of the robots have committed, responses to the commit-messages cease, and 180 seconds later, the robots enter the finished state.

In our earlier discussion, it was pointed out that it was undesirable to have a system working on two mutually exclusive subtasks simultaneously. Global communication permits perfect synchronization of subtask transition, and so a system employing this sort of communication can handle mutual exclusivity with ease. However, because of the stochastic nature of decentralized MRS, it cannot be guaranteed that a system will never split across consecutive tasks, although it is possible to make this phenomenon highly
unlikely. Such overlap could occur since some of the robots will still be in the working state when the first commitment occurs. If any of the committed robots enter the finished state before all of the working robots receive a commit-message, mutual exclusivity will be violated. The probability of this can be reduced by increasing the length of time that robots will remain in the committed state without receiving a response to their commit-messages, but this comes at the cost of a longer delay between subtasks and an increased number of commit-messages sent. We observed zero overlap in any of the communicative systems during the task sequencing behavior, so in our experimental environment, 180 seconds was sufficient.

A. Observed Quorum

The desired quorum for commitment is measured indirectly in the locally communicative systems through the threshold for $q(t)$. This in turn was based on the simplifying assumption that $T_0$ is a constant. Figure 5 plots the observed quorum versus the desired quorum for these trials. The observed quorum was defined as the number of robots in the advocating state at the time that the first robot entered the committed state.

There is a clear linear relationship between the observed quorum and the desired quorum, and this validates the linear relationship illustrated in Figure 3. The slope of the regression line through the data points ideally would be unity, but here it is less than unity. The explanation for this lies in the probability density function (pdf) of the actual inter-robot encounter period. Although the first quartile of this random variable was used for $T_0$, the peak of the pdf occurs at a value less than this. The slope of Figure 5 graph could be increased to unity by selecting a lower value for $T_0$, but caution must be exercised. If $T_0$ is set too low, the threshold $q_{\text{max}}(n_a)$ might be so high that no degree of consensus could induce $q(t)$ reach it.

B. The Cost of Consensus

Our experiments demonstrate that it is possible to specify a desired quorum for a decentralized task sequencing behavior a priori when the robots can communicate with each other, but what is the cost of employing consensus?

<table>
<thead>
<tr>
<th>Communication</th>
<th>Quorum</th>
<th>Time (seconds)</th>
<th>Msgs / Robot</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>n/a</td>
<td>2235.8 ± 199.4</td>
<td>n/a</td>
</tr>
<tr>
<td>Global 1</td>
<td>1</td>
<td>125.4 ± 116.0</td>
<td>0.067</td>
</tr>
<tr>
<td>Global 6</td>
<td>6</td>
<td>560.8 ± 186.6</td>
<td>0.400</td>
</tr>
<tr>
<td>Global 9</td>
<td>9</td>
<td>996.1 ± 314.0</td>
<td>0.600</td>
</tr>
<tr>
<td>Global 12</td>
<td>12</td>
<td>1468.2 ± 405.7</td>
<td>0.800</td>
</tr>
<tr>
<td>Local (1)</td>
<td>1</td>
<td>436.5 ± 179.9</td>
<td>36.5 ± 6.0</td>
</tr>
<tr>
<td>Local (6)</td>
<td>6</td>
<td>666.9 ± 190.4</td>
<td>52.0 ± 5.6</td>
</tr>
<tr>
<td>Local (9)</td>
<td>9</td>
<td>862.1 ± 212.6</td>
<td>65.8 ± 14.0</td>
</tr>
<tr>
<td>Local (12)</td>
<td>12</td>
<td>1291.3 ± 312.2</td>
<td>102.7 ± 23.2</td>
</tr>
</tbody>
</table>

1) Time Costs: Table I presents the time required by each approach to complete the task sequencing measured from the beginning of a trial to the time that the last robot entered the finished state. Clearly the non-communicative system was the most expensive in terms of time costs of all of the configurations investigated. Unless communication is prohibitively expensive and mutually exclusive subtasks are absent, a communicative scheme should be adopted.

The locally communicative robots required approximately 300 additional seconds to complete their task sequencing than their globally communicative counterparts. Figure 6 plots the time of the first commitment (i.e. the time at which a robot first concluded the quorum had been met) versus the observed quorum for both of the communicative systems. The two regression lines are very nearly parallel (slopes of 121.5 and 128.1), so the incremental cost of consensus between the two systems is virtually identical. The additional time required by the locally communicative
consensus estimation is the vertical distance between these lines, which is $\sim 140$ seconds.

The majority of the extra time required by the locally communicative system is spent during the commitment phase, $\sim 250$ seconds. This can be reduced by shortening the time that committed will robots remain in the committed state without receiving a response to a commit-message. However, shortening this period will increase the probability of a system violating mutual exclusivity. The relative importance of task sequencing time versus mutual exclusivity in the mission at hand should guide this tuning.

2) Energy Costs: Energy is consumed by movement and communication. Because the robots always are in motion, the energy cost to each robot due to movement is proportional to the execution time of the task sequencing behavior and its mass [4]. The energy required to transmit each message is proportional to the square of the transmission range, so the total communicative energy cost to each robot is proportional to the number of messages sent times the communication range squared.

For medium and large-sized robots, movement costs dominate. For much smaller robots, the energy requirements of communication can become significant, and even dominate in the case of micro- or nano-robots. In our experimental environment the globally communicative robots had to be able to send messages at least 11.5 meters, while the locally communicative robots had a communication range of 0.65 meters. One global message is thus equivalent to 313 local messages. From the last column of Table I, the locally communicative robots sent only 160-170 times as many messages as their globally communicative counterparts that experienced a similar observed quorum, so our proposed task sequencing behavior is much less expensive from a communications standpoint.

V. CONCLUSIONS AND FUTURE WORK

In this paper, we have presented a behavior-based approach to task sequencing in decentralized systems that requires only very local communication. Our contributions are threefold. First, we introduce the use of an explicit quorum to ensure that a sufficient proportion of a system agrees that it is appropriate to transition to the next task. Second, we developed an elegant technique for the individual robots to compute the degree of consensus in their range of 0.65 meters. Our contributions are threefold. First, we introduce the use of an explicit quorum to ensure that a sufficient proportion of a system agrees that it is appropriate to transition to the next task. Second, we developed an elegant technique for the individual robots to compute the degree of consensus in their range of 0.65 meters. From the last column of Table I, the locally communicative robots sent only 160-170 times as many messages as their globally communicative counterparts that experienced a similar observed quorum, so our proposed task sequencing behavior is much less expensive from a communications standpoint.

\section{A. Future Work}

The manner in which consensus was computed in this work is coupled to the environment through the rate of decay of the exponential quorum function $q(t)$. With some domain specific knowledge, tuning the system is not a problem. However, a domain independent approach is desirable, and this involves the removal of time as a variable. We have recently developed such a technique and are investigating it in simulation. Furthermore, we have constructed a set of robots to analyze the behavior of our task sequencing algorithm in a physical environment.

The contributions of this paper represent a form of group decision-making. The decision made was “Are we done the current task yet?”. Through advocacy and commitment, an emergent behavior was produced whereby the individual members of a decentralized MRS both participated in and adhered to a system-level decision. We are extending the decision-making framework to permit more general group decision-making by decentralized systems.

\section{REFERENCES}


