Development of Top-Down Analysis of Distributed Assembly Tasks

Anthony Cowley
GRASP Laboratory
University of Pennsylvania
acowley@seas.upenn.edu

M. Ani Hsieh
SAS Laboratory
Drexel University
mhsieh1@drexel.edu

C.J. Taylor GRASP Laboratory
University of Pennsylvania
cjtaylor@seas.upenn.edu

ABSTRACT

Distributed assembly tasks, in which large numbers of agents collaborate to produce composite objects out of component parts, require careful algorithm design to ensure behavior that scales well with the numbers of agents and parts. Yet algorithm evaluation, through which design is guided, is complicated by the combinatorial nature of system states over the course of execution. This leads to a situation in which the algorithm design space is often severely cramped by the inefficiency of available analysis techniques. We review several available analysis strategies, and present two techniques for designing distributed algorithms that lend themselves to continuous differential analysis while avoiding catastrophic deviation between discrete and continuous system models. This methodology aims to allow optimization at the macro level to inform parameter choice for discrete, real world systems.

Categories and Subject Descriptors
I.2.11 [Distributed Artificial Intelligence]: Multiagent systems

1. INTRODUCTION

In this work, we describe a flexible manufacturing system based on a robot swarm tasked with assembling composite products from distinct parts. The objective is to develop “top-down” design techniques for decentralized control policies that are invariant to changes in team size and part quantities while satisfying workspace and task constraints. To this end, we consider a distributed assembly task where heterogeneous parts are randomly placed within the environment. Assembly is achieved by tasking robots to wander the workspace, picking up parts as they encounter them, and assembling composite objects when they encounter other robots with complementary parts. The dynamics of the assembly task may be modeled as a chemical reaction network since robot-part and robot-robot interactions can be treated as chemical reactions between different molecules.

Our proposed approach is close in spirit to several previous works in which system dynamics are modeled as chemical reaction networks. Hosokawa et al. used such a model to predict the yield of full assemblies from a collection of vertically stirred modules [3]. Klavins et al. achieved distributed self-assembly from component parts through random collisions of parts that bind and detach from each other based on pre-programmed probabilistic rules [7]. Here, the chemical reaction based model was used to maximize assembly yield by optimizing the spontaneous detachment probabilities of the various components at equilibrium. However, the proposed optimization strategy required the enumeration of all reachable system configurations, which does not scale well with the number of parts. Similarly, Matthey et al. developed stochastic control policies from chemical reaction-based models that enabled a robot swarm to assemble distinct products from a collection of heterogeneous parts [10]. The control policies obtained here provided theoretical guarantees on overall system performance. The use of mobile robots to manipulate and assemble passive parts decentrally is similar to other work [11] where the objective was to derive a rule set to enable the construction of an entire structure out of simple building blocks.

Similar to earlier works by Hsieh et al. [4] and Matthey et al. [10], we propose to develop a “top-down” design methodology for generating stochastic agent-level control policies for a robot swarm based on the mathematical framework used to model chemical reaction networks. Other works have analyzed collective behavior in cooperative robotic tasks [6]. Macroscopic swarm models have been derived to study the performance of a distributed foraging strategy under varying conditions [9], while a similar approach has been used to analyze and study the effects of specialization within large robot teams [5]. In all these works, robots are treated as single molecules and assumed to be capable of simple atomic behaviors, with local interactions between robots governed by a set of reaction rates. Since individual robotic agents can only assume a finite set of basic behaviors, it is possible to model system dynamics solely by considering the population distribution across the set of behaviors. By describing the swarm dynamics via a macroscopic analytical model, these works have shown that it is possible to derive stochastic agent-level control policies to meet a particular desired group-level outcome [1, 4, 10], thus providing a “top-down” versus the traditional “bottom-up” approach to designing group behavior.
We investigate methods to simultaneously adapt the development of these macroscopic analytical models alongside the discrete behavioral algorithms they are to be applied to in order to maximize the fidelity of the models. Improved compatibility between implementation and analysis creates a virtuous cycle where carefully designed algorithms lead to higher fidelity modeling which leads to improved algorithm refinement strategies. Specifically, we consider the execution of collaborative tasks by a swarm of robots whose goal is to assemble composite widgets made of several smaller parts. This is relevant to applications in areas such as flexible manufacturing where it may be desirable to have a system capable of assembling significantly different products on-demand. Other applications include automation of recycling plants and nanoscale assembly where stochasticity is often the norm rather than the exception.

2. PROBLEM FORMULATION

Consider the problem of deploying a swarm of \( N \) robots to assemble complex products from a set of heterogeneous parts. For example, consider the problem of assembling a micro-robot from a pair of wheels, chassis, and a sensor. As such, the set of possible part types is given \( \{A, B, C\} \) where \( A \) corresponds to the sensor, \( B \) corresponds to the chassis, and \( C \) corresponds to the pair of wheels as shown in Figure 1. The assembly of the micro-robot can be broken down into the assembly of intermediate products: either an \( AB \), the attachment of the sensor to the chassis, or a \( BC \), the attachment of the chassis to the wheels, sub-assembly as shown in Figure 2. The sub-assemblies may then be mated with the missing primitive part, either the set of wheels or the sensor, to complete the assembly of the micro-robot as shown in Figure 3. Rather than focus on the details of assembling micro-robots, this work will consider the analogue problem of assembling \( ABC \) widgets since it provides a nice abstraction for more general assembly tasks.

In particular, we assume uniform distributions of each part type, \( A, B, \) and \( C \), within the workspace. Robots navigate the environment by following a trajectory chosen randomly at start-up and upon encountering an environment boundary. As robots wander the workspace, they are tasked to pick-up and assemble intermediate parts, i.e. \( AB \)'s and \( BC \)'s, and/or \( ABC \) widgets as they encounter parts and other robots. For simplicity, we assume that the primitive parts, \( A, B, \) and \( C \) are replaced in the environment as soon as they are picked up for any reason. Furthermore, the intermediate objects \( AB \) and \( BC \) are dropped in the environment upon production, while a successful assembly of an \( ABC \) widget is immediately removed from the workspace, returning each agent involved in its construction to a free state identical to that in which it started. Finally, it is important to note that agents performing these assembly operations have no \textit{a priori} knowledge of their workspace: neither its geometry, the availability of parts, nor the disposition of other agents.

This abstract assembly task requires cooperation between at least two agents, without any high-level coordination. However, while agent-level behaviors that result in cooperative widget assembly are easy to express, and immediately suggest the opportunity for great parallelism through a simple scaling of the number of parts and agents in the environment, system-level performance is contingent on benign interactions between concurrent assembly operations. Toward this goal, we propose to develop robust concurrent assembly strategies that lend themselves to rigorous analysis for the purposes of tuning high-level algorithm parameters. These parameters may include such features as agent-level preference for certain parts in particular situations. Any such biases can have a dramatic effect on system performance, and thus represent important tuning parameters for the system as a whole. Yet, while the effect of such biases at the agent level may be clear from inspection, the effect of their interactions when embodied by hundreds of concurrently operating agents is less clear. In this way, the system tuning process relies on agent-level tuning, and may only be directed by considering multitudes of interacting agents.

To achieve this, we will first develop a baseline approach using a swarm of \( N \) non-communicating robots with limited sensing capabilities. In this baseline case, a free robot discovers and picks up a part by physically bumping into it. Robots encounter each other by a similar physical interaction, at which time they will produce a new composite part if such an assembly is possible given the parts held by each agent.

The second variation is similar to the first, but involves equipping each agent with a sensor that allows for the detection of parts, be they of type \( A, B, C, AB, \) or \( BC \), within some fixed \textit{sensing radius} of the agent. Additionally, each agent is capable of coordinating with any other agent within its \textit{communication radius} in order to perform an assembly operation.

The third variation considered here is one in which agents do not speculatively pick up parts at all. In terms of the
identification scheme of discrete agent states implied by the previous variations, e.g. an agent holding some part of type A, or some part of type C, this variation is as if agents are allowed to exist in several overlapping states simultaneously. An agent may be aware of multiple parts in the environment, e.g. an agent aware of both a part of type A and a part of type C, but does not commit to any subsequent operation until said operation is known to be terminating. That is, when the agent comes into contact with another agent such that the two may combine the parts they are aware of to produce a composite object. Put another way, the previous two problem formulations force an agent to commit to a course of action when it encounters a part: should an agent, upon discovering a part of type A, pick up said part, it has preordained its immediate future to consist of an assembly operation in which it contributes a part of type A.

3. ANALYSIS TECHNIQUES

Given an assembly algorithm defined over a set of parameters, a frequent objective is to determine the optimal set or subset of parameters that can satisfy specific performance metrics, i.e. maximize widget production. Perhaps the most intuitive approach is to search for these optimal parameters by simulating the assembly process. This is most commonly achieved via an agent-based simulation (ABS) where each robot agent is simulated individually, and time is a synchronous signal used to advance each agent’s state of execution simultaneously. While this approach has the advantage of faithfully modeling both the finite individuality of swarm members and the constant advance of time, it ignores many opportunities for improved efficiency. First, if many agents are executing the same behavior, it is not always clear how much is gained by simulating N copies of the agents. Second, the regular sampling of time must be fine enough such that each robot is only expected to be involved in one interaction between simulation samples. Otherwise, the order of events during a time interval is unspecified, which can lead to undefined behavior. However, this fine-grained, regular sampling typically results in many samples when nothing interesting happens. Such intervals are identified by purely deterministic behavior that could be perfectly modeled in a more computationally efficient manner. For instance, the position of a particle moving with constant velocity under the influence of no external forces may be accurately predicted by simply integrating the known velocity, rather than simulating the movement by a sequence of identical discrete jumps through space.

To address the efficiency of time sampling while still explicitly representing the discrete agents that make up the system, one may employ a macro-discrete model [2]. In this model, one arranges to only sample the simulation when an interaction occurs. This is achieved by modeling the rate at which events happen with a stochastic process, typically a Poisson distribution, and advancing the simulation directly between the times at which events occur. A Poisson process with time constant k fires at random times with the firing probability per unit time given by \( k \). The process is Markov since the firing probability is independent of past history. The distribution of intervals between two firings can be derived analytically and is given by \( p(t) = ke^{-kt} \). Thus, one can simulate Poisson transitions in two mathematically equivalent ways. (1) Run iterations with a small time step \( \Delta t << 1/k \); at each iteration, the probability of transition is \( \Delta p = k \Delta t \). The transition is triggered in the current iteration if \( r < \Delta p \), where \( 0 < r < 1 \) is a uniformly distributed random number. This implementation is exact in the limit \( k\Delta t \to 0 \). (2) Generate a random number \( t_r \) distributed according to \( p(t) = e^{-kt} \) and take the transition at time \( t_r \). This second implementation has been shown to be mathematically equivalent to an agent-based simulation [2]. Since each agent is modeled individually, system dynamics dependent on small numbers of individuals may be faithfully captured, while overall simulation performance is greatly increased.

The final strategy considered here is a continuous model of system dynamics. For large enough numbers of robots and parts, it is possible to derive an analytical macroscopic description of the dynamics of the assembly process. Such a model stands in stark contrast to the previous two as it adopts a continuous model of the passage of time, but also takes the drastic step of abstracting discrete parameters into continuously varying values, such as mapping the number of agents engaged in a particular behavior to fractions of the total population. The distinct advantage of this approach is that it brings to bear all the long established analysis tools and techniques from the study of differential equations, and allows for the immediate numerical solution to parameter optimization for most systems. The performance of such methods vastly outpaces the alternate simulation techniques, and thus allows for significantly more rapid iteration of algorithm design since algorithm performance can be easily approximated.

4. ALGORITHM DESIGN

While continuous models are incapable of reflecting the discrete dynamics of a distributed assembly task, they are often good enough, and represent such a compelling performance advantage over other simulation techniques that investigating their applicability is a worthy endeavour. While continuous models treat inherently discrete quantities as continuously varying, their behavior may closely mimic that of the discrete system when large numbers of particles are considered. For example, a continuous model may indicate that, at some point in time, 50.02% of agents are engaged in a particular behavior. If there are one thousand agents in the actual system, then such a configuration is impossible: the assignment of agents to behaviors is entirely discrete. However, if the true performance of the system would have exactly five hundred agents engaged in the behavior, then the deviation of the continuous model from the discrete system may not introduce significant error.

For the specific application of assembly tasks, one may identify the key areas where continuous models break down. One such area is the possibility of deadlock. A deadlocked configuration is one in which no agent is able to change its own state with respect to an assembly task. That is, agents may move about the environment, but none makes any productive contribution to an assembly operation. Such a configuration may result in the example problem described above if every agent should pick up a part of type A, for example, and not release it until it is able to contribute that part to an assembly operation. In this case, no agent is able to find another agent with a complementary part, thus no progress is made in the assembly task. In comparison, a continuous model may suggest that some fraction, say 10%, of a single agent is still free to take productive action, thus
allowing some non-zero probability that the system should recover. Yet the discrete, real system does not allow this configuration: if zero agents are able to make progress, then the system is in a stable state of non-production.

Crucially, such a configuration never occurs in a continuous model of the system. This is because each agent exists in all possible states at once according to some probability distribution; no commitment need be made. This deviation of the continuous model from the real system is catastrophic when it occurs, and arguably invalidates any proposed utility of the entire approach. However, this deviation of the continuous model from real system performance may be precluded by designing the algorithm such that deadlock is not a reachable configuration. An example of such a design technique is a transactional approach to resource-consuming operations.

One may view a productive assembly as a sequence of robot resource acquisitions followed by an assembly operation. Note that the action of a robot picking up a part consumes a robot resource. While it is unknown precisely what assembly operation will ultimately exploit that resource, the robot may retroactively be classified as locked by the abstract assembly operation that eventually uses it. Viewed against time, the interval during which a robot resource is held begins when a free robot encounters a part, and ends when that robot contributes the part to an assembly operation. But this time is unbounded! If instead one is able to bound the time any resource is exclusively held by some operation, then deadlock is avoided. This may be achieved in distributed assembly tasks by taking advantage of the fact that robots are not molecules, and are typically imbued with remote sensing and communication capabilities. Thus, a robot may simply transition between various states of awareness of components in its vicinity without physically acquiring exclusive domain over any one part. Note that the exclusive domain in this case is mutual: the part may be viewed as having exclusive ownership of the robot that has picked it up. The robot resource is consumed by the part until it is able to integrate the part into a composite object.

When a communicating group of robots decides that, collectively, they know how to obtain the components necessary to assemble a composite part, then, and only then, do they move to physically pick up the necessary pieces before engaging in a cooperative assembly operation. Part acquisition may fail, but such an eventuality may now be reasonably detected by a simple time limit on the action of a robot picking up a sensed part. That is, one may assume that a robot actively sensing a part may acquire, or fail to acquire, that part in bounded time. In this way, the entire assembly operation occurs in bounded time, and may be viewed as transactional—the full assembly occurs instantaneously or not at all—by an external observer.

A shortcoming of this approach is that it may be too conservative. In fact, some speculative execution of the assembly task may suffice to overcome an environmental condition such as an excessive sparsity of parts. Specifically, if parts are spaced farther apart than twice the robots’ communication radii, then no two robots will ever be able to share their concurrent knowledge of part locations. This represents a type of live lock, wherein individual robot state, vis-à-vis the set of parts sensed by the robot, changes as time progresses, but no productive work is done. However, if a robot should optimistically acquire a part, thus locking itself to an assembly operation, it may move within communication range of another robot either sensing or holding a complementary part.

In order to avoid deadlock, while maintaining the benefits of optimistic execution, one may specify that some robots act optimistically while other act pessimistically. This amounts to a hedging strategy against unforeseen environmental conditions: the pessimistic strategy is advisable when parts are densely packed as it maximizes potential parallelism, while the optimistic strategy is necessary to make any progress when parts are few and far between. Such a heterogeneous behavior population may be arrived at by programming agents to probabilistically assign themselves one behavior or the other. This stochasticity may be added without affecting the execution of either behavior by wrapping the two deterministic behaviors in a probabilistic conditional expression.

5. EXAMPLE ANALYSIS

One may consider a single probabilistic deterministic algorithm that induces a specific population distribution over discrete classes. This is useful because the single algorithm can be duplicated an arbitrary number of times while always maintaining the desired population statistics. However, for analysis, one can strip off a layer of randomization by representing the algorithm as two distinct sub-populations, each of whose relative prevalence is defined by the statistics of the probabilistic element of the original algorithm. This may be demonstrated by a simple example.

Consider Algorithm 1, which, when executed by a population of $N$ agents equipped with a suitable random number generator approximating a uniform distribution, can be expected to yield $kN$ agents driving to the left, and $(1-k)N$ agents driving to the right. This same algorithm can be deconstructed by lifting the impact of the if...then...else construct into the top-down population specification. Alternatively, this algorithm may be viewed as an implementation of a top-down design directive. Concretely, the chemical kinetics specification,

\[
\begin{align*}
R_{\text{right}} & \xrightarrow{k} R_{\text{left}} \\
R_{\text{left}} & \xrightarrow{1-k} R_{\text{right}}
\end{align*}
\]

may be used to model Algorithm 1, or, from the other direction, the above reaction equations may be implemented at the agent level by Algorithm 1. Here, $\tau$ can be interpreted as a simple scale factor in the time domain. These reaction equations result in a differential model of the population distribution,
In order to evaluate steady state production levels of an assembly algorithm, one would typically be interested in equilibrium conditions of this system,

\[
\begin{align*}
\dot{R}_{\text{left}} &= \frac{k}{\tau} R_{\text{right}} - \frac{1 - k}{\tau} R_{\text{left}}, \\
R_{\text{right}} &= \frac{1 - k}{\tau} R_{\text{left}} - \frac{k}{\tau} R_{\text{right}}.
\end{align*}
\]

The equilibrium condition of the linear system induced by Algorithm 1 indicates the relative proportion of agents driving left, and those that are driving to the right. Given a value for \( k \), \( R_{\text{right}} \) and \( R_{\text{left}} \) can be solved for by imposing a "conservation" condition that \( R_{\text{right}} + R_{\text{left}} = 1 \). That is, the sum of the population fractions must be one. Thus one arrives at the expected result that \( R_{\text{right}} = R_{\text{left}} = 0.5 \) if \( k = 0.5 \), for example. Assembly tasks do not often lend themselves directly to description by linear system, however the resulting system of differential equations may still be solved numerically when no analytic solution is available.

5.1 Results

While the continuous model of the example widget assembly task is free to consider nonsensical concepts such as a fraction of a robot, its performance still matches the agent-based simulation at large population sizes, Figure 4. These graphs show that solutions of the differential equations modeling system dynamics, number of robots holding a part of a particular type and total \( \text{ABC} \) widget production rate, closely match the behavior of the discrete agent based simulation.

The assumption of the existence of partial robots does, however, deviate strongly from reality in the presence of deadlocked behaviors. This is best seen by considering smaller populations executing locking behaviors where deadlocked configurations are statistically probable events. Figure 5 shows the agent-based simulation’s production rate of \( \text{ABC} \) widgets dropping to zero due to the population working itself into a deadlocked configuration. In this scenario, the most common cause for deadlock proved to be too many agents holding onto intermediate parts of type \( \text{AB} \) or \( \text{BC} \) while the remaining agents held parts of type \( \text{B} \). If all the agents in the system assume one of those roles, then overall production ceases. The cessation of production is stable and unrecoverable: the controller makes no allowance for detecting and escaping a system-wide deadlocked behavior.

This type of deadlock, where all robot resources are consumed, may be avoided by a transactional assembly approach that does not speculatively lock a robot to a particular part. The steady state production rate for the transactional assembly technique, as seen in Figure 6, dwarfs the steady state of the locking approach shown in Figure 4, even when just considering the continuous model. This is due to the greater extent to which the transactional assembly behavior exploits potential concurrency in the system: the duration for which a robot resource is exclusively locked is strictly bounded. Turning to the agent-based simulation results shown in Figures 4 and 6, the danger of deadlock in the speculative locking strategy is starkly apparent. While the continuous model is not susceptible to falling into a stable “stuck” configuration, the discrete system is, and may do so at any time.

However, the conservative approach taken by the transactional assembly behavior has its own drawbacks. While it avoids prematurely locking a robot resource, it is far more susceptible to production shortfalls due to unfortunate environmental conditions. Namely, if part density is low, there is a chance that individual parts may be so widely spaced that two robots can never simultaneously sense two different, compatible parts and be in communication range of each other. This occurrence is, once again, not seen in the continuous model since, in such a model, a low part density means that a productive configuration, in which two robots sense two parts and can communicate, is unlikely, but never impossible. In a discrete system, however, the transactional assembly behavior may yield a production rate of zero if the parts are distributed with too much distance between them. This second form of deadlock, in which no positive progress may be made by the system, can be avoided by a mixed population.

Without complicating agent-level behaviors, one may avoid both forms of deadlock by hedging against either a sparsity of robots or a sparsity of parts. The performance advantage
of a mixed population is shown in Figure 7, in which 100 robots are operating in an environment with an uneven part distribution. Namely, there are regions of the environment densely packed with parts, and there are regions with very low part density. The production rate of the transactional assembly behavior, shown with a solid red line, plummets when a significant fraction of agents find themselves in low density regions, while the heterogeneous team has a much higher minimum production rate.

6. DISCUSSION

One can see, by comparing Figure 6 with Figure 5, the difference in absolute widget production rates for a given set of system parameters. Even if one ignores absolute productivity measures, the deadlock-freedom of the transactional assembly behavior trumps any other performance considerations of different agent-level behaviors.

Modifications to agent-level behavior that avoid deadlocked configurations have typically been implemented as spontaneous decay reactions in other robot swarm simulations based on chemical reaction networks [8, 7, 5, 10]. However, such spontaneous reactions, in which a robot has some non-zero probability of simply dropping a part it is carrying, may not be necessary in order to avoid deadlock configurations. Instead, deadlock freedom may be attained by more deterministic behavior specification, as in the transactional assembly scheme that avoids robot starvation.

By working with agent-level behaviors that preclude deadlocked scenarios, one is able to leverage computationally efficient differential models of system dynamics. This ability means that one can quickly predict system performance for a given set of parameters, e.g. part density, and take some action to adjust these parameters if system performance is insufficient. Hedging strategies, as demonstrated in the mixed transactional-speculative population above, may be efficiently implemented by a probabilistic role assignment mechanism in which agent’s adopt a particular behavior based on a desired distribution. This assignment technique is robust to changes in population size, and requires no per-agent customization, thus making it suitable for swarm deployment scenarios. The probabilistic combination of deterministic behaviors represents a sweet spot of easily understood agent-level behaviors, scalability to large, varying...
population sizes, and amenability to differential modeling techniques.

7. REFERENCES


