

Forthcoming in Proceedings of the Tenth International Workshop on Multi-Agent-Based Simulation (MABS 2009), Budapest, Hungary, May 11-12, 2009.

Stigmergic Modeling of Hierarchical Task Networks

H. V. Parunak, T. Belding, R. Bisson, S. Brueckner,
E. Downs, R. Hilscher

Keith S. Decker

Vector Research Center, TTGSI
3520 Green Court, Suite 250
Ann Arbor, MI 48105 USA
{van.parunak, ted.belding, robert.bisson,
sven.brueckner, liz.downs,
rainer.hilscher}@newvectors.net

University of Delaware
444 Smith Hall
Newark, DE 19716 USA
decker@cis.udel.edu

Abstract. Stigmergy is usually used to model semantically simple problems such as routing. It can be applied to more complex problems by encoding them in the stigmergic environment. We demonstrate this approach by showing how stigmergic agents can plan over a hierarchical task network, specifically a resource-oriented dialect of the TÆMS language.

Keywords: Stigmergy, HTN, TÆMS, Planning, Scheduling, Interaction

1 Introduction¹

Stigmergy, in which agents coordinate actions by making and sensing changes to a shared environment, is often applied to routing problems. Agents deposit digital markers analogous to insect pheromones in the environment. The environment aggregates deposits from different agents (fusing information), propagates them to nearby locations (coupling local actions with global objectives), and evaporates them (discarding obsolete information). Agents base decisions on a function of nearby pheromone strengths. For example, an agent representing a network packet may climb a pheromone gradient to find the most efficient path to its destination.

The semantics of route planning are fairly constrained, and stigmergy is usually considered a low-level form of cognition [13]. But this semantic constraint really says more about the nature of a flow network than of stigmergic interaction. One can imagine using stigmergy to coordinate agent behaviors on a network that embeds much more complex semantics. Stigmergy can transfer cognition from the agents to the environment. A cognitively rich environment can yield cognitively complex outcomes among relatively simple agents, as in Simon parable of the ant [15].

Higher cognition is usually considered necessary for coordinated execution of complex tasks. For example, the treatment plan for a hospital patient has both internal relationships (some tests must be done in a particular order or within certain time

¹ This research was conducted with the support of the office of Naval Research (Contract # N00014-06-1-0467). The results presented do not necessarily reflect the opinion of the sponsor.

limits) and external relationships between treatment plans (only one MRI machine exists; certain ancillary hospital units prefer to run similar tests in batches to reduce set-up times, etc.) [5]. Another example is the coordination of pre-planned activities in dynamic environments such as military, law-enforcement, or disaster planning scenarios [3]. Several law-enforcement units may wish to surprise suspects simultaneously at different locations so they cannot warn each other. Besides coordinating the surprise itself, some units may require equipment or information whose delivery time is not known in advance. The structure of such tasks can be represented as a graph, specifically, a hierarchical task network or HTN.

The usual approach to such scenarios is to give complex agents an internal representation of their own plans (and how they relate to the plans of other agents). Examples include CSC agents [7], or unrolling each agent's view of the HTN into a Markov decision process [8], or translating it into a Simple Temporal Network [16]. We take a different approach. Rather than putting the HTN inside complex agents, we put stigmergic agents inside the HTN. Coordination is achieved, not by dialogs based on each agent's individual analysis of the HTN, but by means of interactions among the agents mediated by the structure of the HTN itself.

The motivation for this alternative approach to reasoning over HTN's is twofold. Theoretically, it shows how stigmergy can be applied to cognitively complex problems. Practically, by turning the problem inside out, stigmergy permits the analysis of HTN's that are much too large to be analyzed using classical approaches.

This paper demonstrates stigmergy on an HTN, in a dialect of the TÆMS task language [6]. Section 2 reviews TÆMS and the rTÆMS dialect, which emphasizes the importance of resources in coordination. Section 3 shows how to apply stigmergy to an rTÆMS graph (along the way analyzing some "obvious" approaches that do not work). Section 4 reports experiments that demonstrate this approach for single-agent planning. Section 5 discusses the relative virtues of stigmergic and more complex agents in reasoning over HTN's, and outlines future work. Section 6 concludes.

2 TÆMS and rTÆMS

A hierarchical task network (HTN) is a collection of events, with two kinds of relations among them: a hierarchical structure relating tasks to subtasks, and constraints on the order of execution among the tasks.

2.1 Introduction to TÆMS

Figure 1 illustrates TÆMS on the dining philosophers.

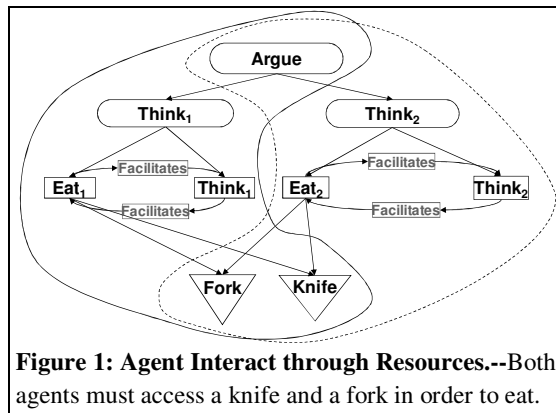


Figure 1: Agent Interact through Resources.--Both agents must access a knife and a fork in order to eat.

- Ovals (“Argue,” “Think₁”) are *tasks* and *subtasks*. Tasks and subtasks may be associated with one or many agents, and can be further subdivided.
 - Rectangles (“Eat₁”) are *methods*, which are the lowest level of activity. Each method is associated with a single agent.
 - Labeled arcs between methods (“Facilitates”) are *non-local effects* (NLE’s), which capture precedence constraints.
 - Inverted triangles (“Forks”) are *resources*, which methods produce and consume.
- Curved lines show the subgraphs accessible to each agent, their *subjective* graphs. The overall graph, describing the complete problem, is the *objective* graph.

A method’s execution produces quality that flows up to its dominant subtasks and tasks. Each task or subtask has a Quality Accumulation Function (QAF), providing a more nuanced way to capture what other HTN’s represent as AND and OR branches. For example, a Min QAF says that the quality of a task is the minimum of the incoming qualities, and thus remains at 0 until all subtasks or methods execute (an AND), while a Max QAF corresponds to an OR, yielding nonzero task quality as soon as any subtask succeeds. More complex functions are also possible.

The version of TÆMS in [6] explicitly represents dependencies between events and resources. C_TAEMS [1] does not represent resources.

Sometimes we need a more general vocabulary. Tasks, subtasks, methods, and resources are all *nodes* in a graph, whose *relations* are provided by non-local effects, resource dependencies, and QAF’s. Tasks, subtasks, and methods all describe *events*.

Because HTN’s are graphs, and because graphs are convenient stigmergic environments, could we use an HTN directly as an environment for stigmergic interaction? Perhaps the events in an HTN could serve as places of our environment where agents deposit and sense digital pheromones. On reflection, this approach poses a problem. Commonly, a single agent is responsible for each event (or at least each method), so different agents cannot interact through a single method.

However, even private events must access shared resources. So it is natural to use resources as the basis for coordination. Competition for resources is a form of stigmergic coordination [12], suggesting that resources are natural candidates for the places of a stigmergic environment.

2.2 rTÆMS (resource TÆMS)

The intuition in the previous section is useful only if there are enough relations in a structure to mediate all the events that need to be coordinated. In [6], resources are only involved in some task relationships. Subtasking relations and NLE’s do not involve resources, and [1] ignores them altogether.

But things that behave like resources are ubiquitous in TÆMS structures, and lie behind both subtasking and NLE’s. We argue that a fully elaborated TÆMS structure is a bipartite graph, whose two components are events and resources.

Quality as a Resource.—Quality is central in TÆMS. It is produced by every event, and determines the sequencing of events. This behavior is reminiscent of a manufacturing job shop, where operations add successive features to parts that move between them. One operation might cut a bar of material to length, the next might drill a hole in it, and the next might tap the hole. TÆMS events are analogous to manufacturing operations, and TÆMS quality is analogous to features.

In a manufacturing system, the parts that move from one operation to the other are naturally modeled as resources, produced by one operation and consumed by the next. The actual resource would not be the bare part, but part-with-specified-features. Perhaps we can model quality as a (virtual) resource and capture its effects through resource relations. To verify this hypothesis, we need to explain how quality-as-resource can accommodate non-local effects (NLE's), the subtasking relation, and QAF's. We summarize this mapping. For a complete description, see [9].

Non-Local Effects (NLE's).—C-TÆMS defines four NLE's: *Enables*, *Facilitates*, *Disables*, and *Hinders*. Each has a source, a destination, and a delay. *Facilitates* and *Hinders* also have discrete distributions over quality, cost, and duration. The semantics of NLE's assume that the destination is a method, though they are often drawn between tasks as shorthand. We map each of these into a NLE method. This method has a duration equal to the specified delay, is limited by the quality produced by its source, and at the end of its duration,² produces the same amount of quality as its input, and makes it available to its target. A NLE method requires some computational element to execute it. The natural candidate is the environment [17].

QAF's.—Some QAF's [6] (e.g., SeqMax, SeqMin) constrain sequencing among subtasks. These have disappeared in [1], and can always be handled using the apparatus of NLE's outlined above. The QAF's that remain are Sum, Max, Min, SyncSum, SumAnd, and ExactlyOne.

Subtasks produce their individual qualities, and sometimes additional resources. The supertask then combines individual qualities. The Task computes its quality by applying the appropriate function (Max, Min, Sum) to the qualities of its subtasks.

3 STIGMERGY OVER rTÆMS

This section summarizes stigmergy, analyzes two approaches to applying it to HTN's that did not succeed, then describes one that does.

3.1 Basic Stigmergic Approach

Our approach is based on polyagents [11], which represent each entity in the real world with a set of agents. A single persistent avatar sends out a stream of ghosts that explore alternate routes through the environment, recording their experiences by constructing pheromone fields over that environment. These fields allow the ghosts to interact with the possible futures of other entities, exploring a much richer set of potential interactions than is possible with single-trajectory simulations. The avatar picks its next step by consulting the pheromone fields generated by its ghosts.

² This is the DTT specification favored by Decker [4]. An alternative specification permits quality to accumulate as a method executes.

3.2 The Resource Dual of a TÆMS Graph

A network of tasks and methods is a poor stigmergic environment because at least the methods, and often the tasks, belong to single agents.

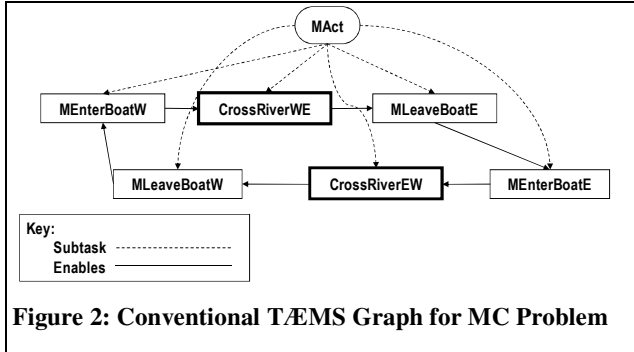


Figure 2: Conventional TÆMS Graph for MC Problem

Resources, not tasks, are the locus of most coordination. By considering quality as a resource, we can represent any TÆMS graph as a bipartite graph whose nodes are resources and events.

It is possible to construct the resource dual of a TÆMS graph, containing only resources. Figure 2 exhibits a resource-free TÆMS graph for a Missionary in the Missionary-Cannibal problem, in the spirit of [10]. Relation types are encoded by line patterns (dashed for subtasks, solid for Enables). The bold methods are public methods, those in which both Missionaries and Cannibals participate jointly. The technical memo [9] elaborates this structure into a full rTÆMS graph, then derives the resource dual, a hypergraph shown in Figure 3. In this graph,

- The outer ring of qXXXX quality resources captures the same structure as the ring of events in Figure 2.
- The structure among the population resources (mX, cX) and between them and the quality resources capture important constraints on enabled events.
- Eight nodes are shared with other agents, offering a richer stigmergic environment than the two shared nodes in Figure 2.
- Paths of public nodes (bold outlines) connect every pair of private nodes, providing paths through which agents can constrain one another's behavior.

In spite of its promise, we were unable to solve the missionary-cannibal problem by swarming on Figure 3. By omitting events from the agents' environment, we lost important distinctions among different ways

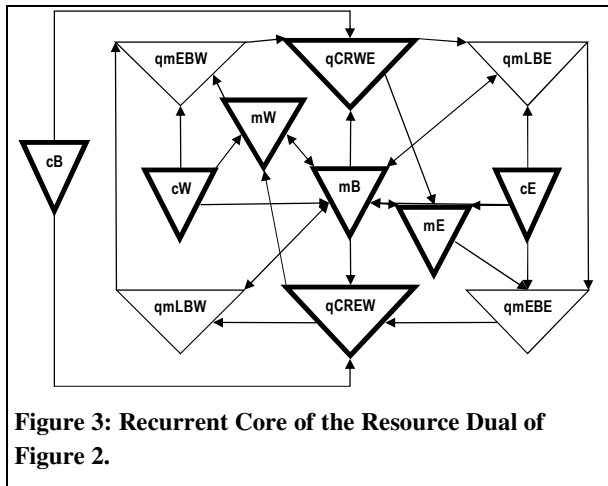


Figure 3: Recurrent Core of the Resource Dual of Figure 2.

that resources could be replenished or consumed, leaving the ghosts with too little information to solve the problem.

3.3 Full rTÆMS Graph

Next we swarmed on a full rTÆMS graph. Ghosts deposit and sense pheromones on both resources and events. Pheromone on a *resource* enables the methods that require that resource. Pheromone on a *method* tells agents how desirable the method is for execution. Even if the method is enabled (that is, its resources have pheromones), an agent might still prefer an alternative method. As ghosts move out from their avatar to explore alternative futures, they deposit pheromone on the resources they encounter to enable the execution to unfold. On the way back, having reached the end of their exploration and evaluated the outcome, they deposit pheromone on methods to show the desirability of the path they followed. This pheromone field then guides not only other ghosts, but also the avatar when the time comes to execute the plan.

This approach was also unsuccessful. The dominant structure of the graph is a hierarchy. Ghosts moved up and down the hierarchy in order to move from one method to another. In the process, they bottlenecked at branch points in the hierarchy.

An HTN's hierarchy shows how methods and subtasks are organized into tasks, but does not highlight the causal flow. In a hierarchy, node proximity represents the logical structure of the task, not its temporal structure. In solving a planning or scheduling problem, agents need to attend to the temporal relations among events. The most important nodes for them to access at any moment are not superordinate and subordinate tasks, but the next nodes in the causal sequence. An entity executing a process is always on one method, looking for the next to visit. The entity's movement is from method to method, not from method to subtask to task and back down. Bottlenecking at branch points is a symptom of the inappropriateness of the hierarchical topology to the problem we are trying to solve.

3.4 Quality Graph vs. Execution Graph

Our current approach decomposes the graph into two parts, with different functions. We discuss first the decomposition, then how the avatars and ghosts explore it.

3.4.1 The Decomposed rTÆMS Graph

Any rTÆMS graph includes an execution graph and a quality hierarchy (Figure 4).

The methods and the resources (physical or virtual) that constrain their sequencing form the *Execution Graph*. Polyagents live on the execution graph, not on the rTÆMS hierarchy. For clarity, the execution graph in the figure is incomplete, omitting the constraints implied by the two virtual resources higher in the hierarchy. In practice, we compile the rTÆMS graph to generate a complete activity graph. Current TÆMS dogma is that a method can be executed only once. An agent that internalizes the graph instantiates new schemata as needed, and in this case the execution graph will be a DAG. It is more natural for agents that live within the graph to revisit a node, and

we intend to support reentrant methods, since some problems (notably Missionary and Cannibals) cannot be solved with a bounded set of methods.

The subtask hierarchy computes and communicates the quality achieved by the system, so we call it the *Quality Hierarchy*.

Compilation of the rTÆMS graph yields two functions that utilize this hierarchy.

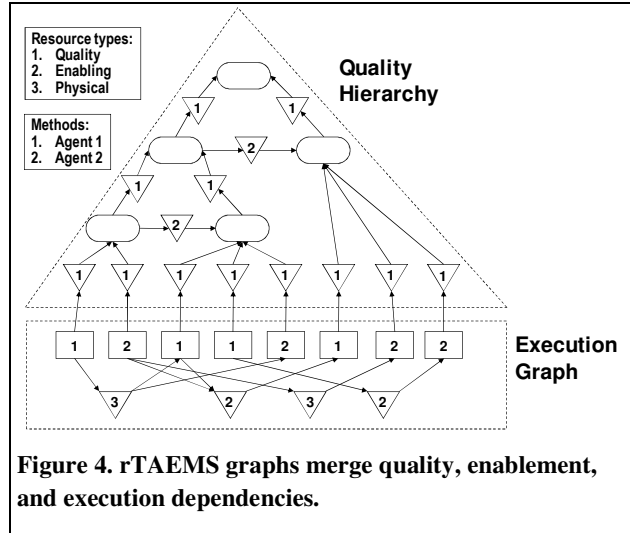
One function propagates quality up the graph, keeping all subtasks notified of their current state of quality. This might support a human-meaningful interface, perhaps by shading each subtask with its current level of quality.

The other function uses the current state of quality to propagate method desirability back down the graph. The desirability of a method depends on its Quality Improvement Potential (QuIP), how much difference to the overall mission the quality it would produce would yield. For example, if two methods are OR'd into a subtask, the more quality has already accumulated at the subtask, the less difference additional quality would make, and the less desirable the execution of the methods is.

The distinction between the Execution Graph and the Quality Hierarchy (with its associated functions) illustrates two important functions of the environment in stigmergic reasoning. The environment *localizes* agents and their effects to reduce their computational burden. In addition, it is *active*, offloading some computation from the agents. (In routing systems, the aggregation, propagation, and evaporation of pheromones are examples of environmental action.) In this case, the Execution Graph provides meaningful localization by ensuring that an agent's neighborhood includes the most relevant methods for it to consider next, while the functions of the Quality Hierarchy extend the environment's actions to support the planning task.

3.4.2 Searching the Execution Graph

An entity's polyagent seeks an optimal path through the Execution Graph. If multiple entities execute concurrently in the same Execution Graph, their stigmergic interaction via shared resources will result in individual plans that are optimized conditional on one another, in other words, a coordinated plan. Our algorithm resembles the polyagent algorithm for manufacturing in [2], but with the addition of quality feedback from the Quality Hierarchy.



The polyagents use three pheromone flavors to coordinate their search for an optimal path through the Execution Graph. Avatars deposit *Execution* pheromone on methods that they have executed, to indicate that they do not need to be executed. (To handle the reentrant case mentioned above, we allow this pheromone to evaporate.) Ghosts deposit *Exploration* pheromone on methods as they visit them. In addition, after they complete their trajectories, they deposit *Desirability* pheromone proportional to the overall quality that they have achieved on all the nodes that they visited. Methods propagate both Execution and Exploration pheromones to the resources that they provision.

Each time a ghost is activated, it chooses among 1) sleep, 2) choose a new method and execute it, or 3) report and terminate.

The *sleep* behavior, chosen by flip of a coin, allows the system to explore alternative orderings of method execution.

If a ghost's lifetime is not exhausted, it *chooses and activates a method*. It identifies the subset of its avatar's methods with no Execution or Exploration Pheromone, but with Execution or Exploration Pheromone on each incoming resource. It scores each such method with a weighted sum of the Desirability Pheromone deposited on that method by previous ghosts, and the quality improvement potential (QuIP) propagated to the node by the quality hierarchy. The QuIP represents the increase in quality that would be realized *at the root of the tree* if the method were chosen. Then it selects a node with a roulette wheel whose segments are weighted by the scores of each method. It moves to the node, and sleeps for a period of time corresponding to the node's execution duration.

If a ghost's lifetime is exhausted, or if there are no methods left for it to visit, it assesses the quality of its overall trajectory, *reports* this quality by depositing Desirability Pheromone on each method in its trajectory, and *terminates*.

At any moment, an avatar is situated on some method in the execution graph, modeling the execution of that method. When execution is complete, it deposits Execution Pheromone on the method to indicate that it has been executed. Then it selects from those methods that do not yet have Execution Pheromone, based on the Desirability Pheromone deposited by the ghosts.

In this approach, the Execution Graph provides agents with a topology in which nearby nodes are also the most relevant ones to the execution of the process in its current state, while the Quality Hierarchy uses the overall task structure to update the desirability of each method based on the current quality of the higher-level tasks.

4 EXPERIMENTAL PERFORMANCE

We demonstrate our method on the rTÆMS graph of Figure 5. This graph does not represent any specific problem, but is constructed to capture two different kinds of constraints that can emerge in a HTN: precedence constraints among methods (captured in the Execution Graph), and the subtask structure represented by the Quality Hierarchy. For clarity, we omit the resources that in fact occupy each link.

We compare our algorithm to two simpler algorithms, on Figure 5 and two simpler graphs, for $3 \times 3 = 9$ experimental configurations, each replicated 25 times.

The three algorithms are:

A1. A random baseline selects methods randomly, ignoring both their enablement in the Execution Graph and their contribution of quality root in the Quality Hierarchy. If a method chosen is not in fact enabled, it remains eligible for later selection. If it is enabled, it is executed and removed from the pool of methods, and other methods dependent on it are enabled. Thus the random method can take arbitrarily many steps to select all methods. In the mean field limit, this process can be modeled as a cumulative advantage process [14], a form of preferential attachment, but the probabilities involved in fact change as selections take place, due to the nonuniform nature of precedence links among the methods.

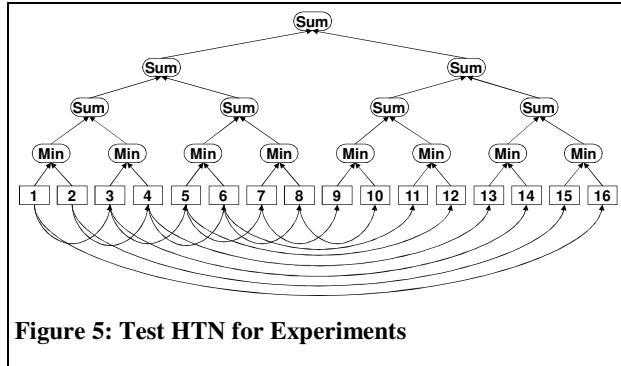


Figure 5: Test HTN for Experiments

A2. A version of our algorithm that ignores QuIP and selects methods based only on enablement and desirability, yielding a process similar to that described in [2].

A3. Our full algorithm, with selection among enabled methods based on QuIP.

The three versions of the network are:

N1. A single task from which all methods descend directly, with no physical resources or non-local effects among them.

N2. N1 with the addition of precedence constraints among the methods.

N3. N2 with the addition of subtasks (and associated Quality Accumulation Functions) between the root task and the methods.

We monitor two dependent variables: how rapidly quality accumulates, and the variation among different runs. Figure 6 shows quality vs. time for each configuration. The maximum quality available is 16 units (one for each method), except in N3, where the use of *Min* QAF's reduces it to 8. A2 and A3 reach maximum quality in 16 steps. A1 does so only in N1. Its average time to reach maximum quality in N2 is 47.3 ± 12.6 steps, and in N3, the statistically identical 46.4 ± 14.8 steps.

The rate of quality accumulation shows a strong correlation between the nature of the network and the effectiveness of the various algorithms. When the network is unconstrained (N1), all algorithms perform the same. Addition of precedence constraints gives A2 and A3 an advantage over A1, but they perform comparably to one another. When we introduce considerations of quality accumulation imposed by the Quality Hierarchy, A3 accumulates quality more rapidly than the other two.

This result highlights two different kinds of complexity in an HTN, one due to precedence constraints among methods, the other to the task structure through which methods deliver their quality. One could characterize HTN's by their location in this two-dimensional space, and distinguish solution strategies by the dimension(s) they address. This distinction, obvious from our inside-out approach to HTN's, is likely to be important for classical AI reasoners as well.

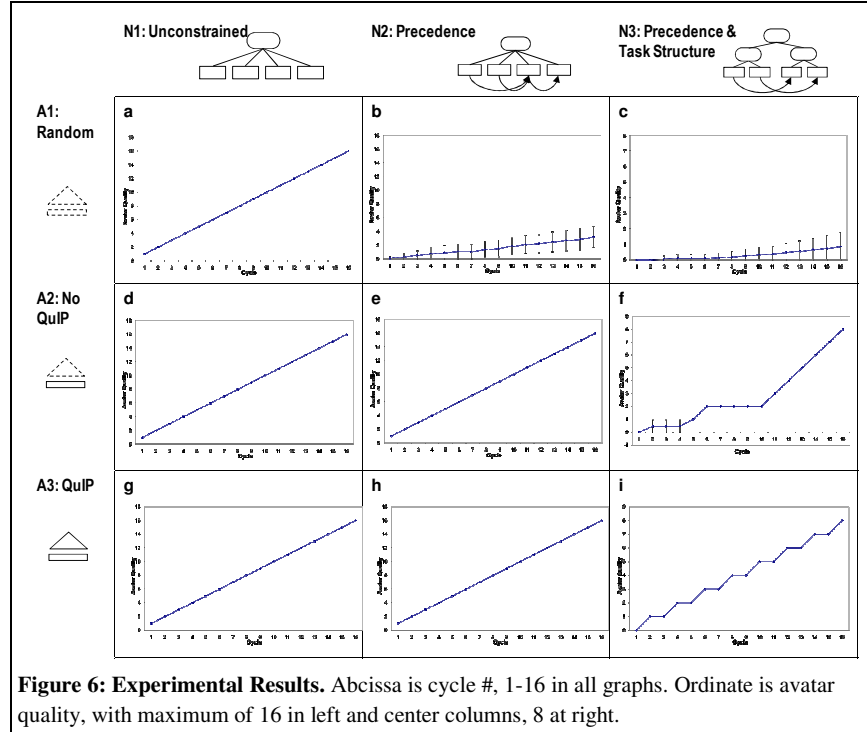


Figure 6: Experimental Results. Abcissa is cycle #, 1-16 in all graphs. Ordinate is avatar quality, with maximum of 16 in left and center columns, 8 at right.

Now consider the variance in the various graphs. As problem complexity increases, the more sophisticated algorithm has less variance. This difference is an important advantage for our methods. Each run of the system represents an actual execution of the problem by a physical entity guided by an avatar, which does its planning guided by swarming ghosts. Real-world agents do not have the luxury of solving the problem 25 times and taking the best result, and high variance means that the simpler methods impose a risk of unacceptably low performance.

5 NEXT STEPS

We are planning to extend this work in a number of ways.

To validate the claim of execution efficiency made in the introduction, we are executing our algorithm on larger graphs, and on scenarios for which benchmark results from classical approaches are available.

Our current experiments focus on a single avatar. We will explore coordination among multiple polyagents.

The methods in our current experiments do not include several features that are needed in solving realistic problems. These features, which can be accommodated with straightforward extensions, include deadlines, task quality that degrades with the

passage of time, a quality threshold below which no quality is generated, reentrant methods, conservation of physical resources, and respect for resource capacity limits.

Stigmergy in general, and polyagent simulation of multiple futures in particular, are heuristics. Like all heuristics, they need to balance simplification against accuracy of results. One simplification that invites more careful study has to do with the ergodicity of causal sampling. Consider two methods, one of which (A) generates a resource needed by another (B). Under our current system, the future explored by one ghost might visit A, thus provisioning its resource and enabling B, but it might not visit B. Another ghost finds B enabled and visits it, but never visits A. Neither future is in fact feasible for the avatar. Our current approach is based on the hypothesis that averaged over a large number of ghosts, such anomalies will be dominated by valid futures, but more detailed experimentation is needed to verify this. The problem can be addressed by ghosts that make more careful use of the trajectory stacks that they are already building, but at the expense of their execution speed.

6 CONCLUSION

Stigmergy can be applied to semantically complex problems (such as planning) by embodying the semantics in the environment over which the agents swarm. It offers significant advantages over methods that apply more complex reasoning methods to the entire plan representation. For one thing, stigmergic processes can be scaled by distributing them over multiple processors. For another, conventional approaches encode specific constraints (NLEs or QAF's) in heuristics, and are brittle in the face of introducing new classes of constraints, but the rTÆMS vocabulary of virtual resources can easily be configured to capture any constraint as a graph structure over which our agents swarm naturally.

Our exploration of this approach on HTN's has led to several important lessons.

- A representation must have locations that are shared among agents in order to support stigmergy. A TÆMS graph with only physical resources, or (as in C-TÆMS) with no explicit resources, does not satisfy this requirement, but can be made to do so by adding virtual resources to form a bipartite graph.
- Though shared nodes (in this case, resources) are necessary to support stigmergy, they are not sufficient. The information in the rest of the graph is also needed.
- Having the agents swarm over all kinds of nodes is not necessarily the best way to give them this information, as our experiments in Section 3.3 showed. Some parts of a graphical representation of a domain may be useful to localize the agents, while other parts can support actions on the part of the environment.
- The topological neighborhood of the agents must be relevant to the task that the agents need to perform.
- HTN's exhibit two qualitatively different kinds of complexity (reflected in precedence constraints and subtask hierarchy), which yield to different aspects of our algorithm. This distinction is probably relevant for other HTN reasoners as well, and methods to characterize an arbitrary HTN along these two dimensions would be useful in selecting appropriate solution tactics.

7 REFERENCES

1. Boddy, M., Horling, B., Phelps, J., Goldman, R.P., Vincent, R., Long, A.C., Kohout, B., Maheswaran, R.: C_TAEMS Language Specification, Version 2.02. DARPA, Arlington, VA (2006)
2. Brueckner, S.: Return from the Ant: Synthetic Ecosystems for Manufacturing Control. Thesis at Humboldt University Berlin, Department of Computer Science (2000)
3. Chen, W., Decker, K.: The Analysis of Coordination in an Information System Application--Emergency Medical Services. In Bresciani, P., Giorgini, P., Henderson-Sellers, B., Winiko, M., Editors, Agent-Oriented Information Systems, vol. 3508, LNAI, pages 36-51. Springer, New York, NY (2005)
4. Decker, K.: Environment Centered Analysis and Design of Coordination Mechanisms. Thesis at University of Massachusetts, Department of Computer Science (1995)
5. Decker, K., Li, J.: Coordinating mutually exclusive resources using GPGP. *Journal of Autonomous Agents and Multi-Agent Systems*, 3(2):133-158 (2000)
6. Horling, B., Lesser, V., Vincent, R., Wagner, T., Raja, A., Zhang, S., Decker, K., Garvey, A.: The Taems White Paper. Multi-Agent Systems Lab, University of Massachusetts, Amherst, MA (2004). <http://dis.cs.umass.edu/research/taems/white/>
7. Maheswaran, R.T., Szekely, P., Becker, M., Fitzpatrick, S., Gati, G., Jin, J., Neches, R., Noori, N., Rogers, C., Sanchez, R., Smyth, K., Vanbuskirk, C.: Predictability & Criticality Metrics for Coordination in Complex Environments. In Proceedings of Seventh International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2008), (2008)
8. Musliner, D.J., Durfee, E.H., JianhuiWu, Dolgov, D.A., Goldman, R.P., Boddy, M.S.: Coordinated Plan Management Using Multiagent MDPs. In Proceedings of AAAI Spring Symposium on Distributed Plan and Schedule Management, (2006)
9. Parunak, H.V.D.: HTN-Induced Stigmergic Environments. NewVectors division of TTGSI, Ann Arbor, MI (2008)
10. Parunak, H.V.D., Brueckner, S.: Ant-Like Missionaries and Cannibals: Synthetic Pheromones for Distributed Motion Control. In Proceedings of Fourth International Conference on Autonomous Agents (Agents 2000), pages 467-474, (2000)
11. Parunak, H.V.D., Brueckner, S.: Concurrent Modeling of Alternative Worlds with Polyagents. In Proceedings of the Seventh International Workshop on Multi-Agent-Based Simulation (MABS06, at AAMAS06), Springer (2006)
12. Parunak, H.V.D., Brueckner, S., Fleischer, M., Odell, J.: A Preliminary Taxonomy of Multi-Agent Activity. In Proceedings of Autonomous Agents and Multi-Agent Systems (AAMAS 2003), pages 1090-1091, ACM (2003)
13. Parunak, H.V.D., Nielsen, P.E., Brueckner, S., Alonso, R.: Hybrid Multi-Agent Systems. In Proceedings of the Fourth International Workshop on Engineering Self-Organizing Systems (ESOA'06), Springer (2006)
14. Price, D.d.S.: A General Theory of Bibliometric and Other Cumulative Advantage Processes. *Journal of the American Society for Information Science*, 27(5-6):292-306 (1976)
15. Simon, H.A.: *The Sciences of the Artificial*. Cambridge, MA, MIT Press (1969)
16. Smith, S.F., Gallagher, A., Zimmerman, T., Barbulescu, L., Rubinstein, Z.: Distributed Management of Flexible Times Schedules. In Proceedings of Sixth International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2007), pages 472-479, IFAAMAS (2007)
17. Weyns, D., Parunak, H.V.D., Michel, F., Holvoet, T., Ferber, J.: Multiagent Systems, State-of-the-Art and Research Challenges. In Proceedings of Workshop on Environments for Multi-Agent Systems (E4MAS 2004), pages 1-47, Springer (2004)