

Stigmergic Reasoning over Hierarchical Task Networks

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

ABSTRACT

Stigmergy is usually associated with semantically simple problems such as routing. It can be applied to more complex problems by encoding them in the environment through which stigmergic agents interact. 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. We not only report our successful method, but also analyze a number of unsuccessful attempts. Our results reveal an important distinction among HTN's.

Categories and Subject Descriptors

I.2.11 [Distributed Artificial Intelligence]: Multiagent systems;
I.2.8 [Problem Solving, Control Methods, and Search]: Plan execution, formation, and generation; Scheduling.

General Terms

Algorithms, Experimentation.

Keywords

Stigmergy, TÆMS, Planning, Scheduling, Biology, Agents, Interaction

1. INTRODUCTION

Stigmergy, in which agents coordinate their actions by making and sensing changes to a shared environment [7, 11, 16], is a powerful mechanism for dealing with many complex problems, including optimization [6], shop-floor control [2], air-traffic control [20], telecommunications [21], and battle prediction [13]. In such systems, agents are situated in the environment, and their interactions are local with respect to the environment, thus reducing the computational complexity that they must face. Dynamics in the environment map between these local agent interactions and global problem parameters. The approach is well suited to very simple agents, such as digital analogs of ants, but in fact even humans use varieties of stigmergy in their daily interactions [12], suggesting its potential for more complex applications.

The most common applications of stigmergy solve routing problems, often over a topological manifold or a flow network. Typically, agents deposit digital markers analogous to insect pheromones in the environment. The environment's dynamics consist of aggregating the deposits from different agents (a form of information fusion), propagating them to nearby locations (enabling

the mapping between local actions and global objectives), and evaporating them (discarding obsolete information, and thus solving the truth maintenance problem in constant time). Agents respond to the resulting pheromone field by basing their decisions on some function of the pheromone strengths in their local neighborhood. For example, an agent representing a packet in a digital network may climb the pheromone gradient to find the most efficient path to its destination.

The semantics of route planning are fairly constrained, and stigmergy is usually applied as a fairly low-level form of cognition [18]. 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. The genius of stigmergy is in transferring cognition from the agents to the environment. A cognitively rich environment can yield cognitively complex outcomes among relatively simple agents, as Simon argued long ago in his parable of an ant on the beach [22].

One domain for which higher levels of cognition are often considered necessary is coordination in the execution of complex tasks. For example, tests and treatments on a hospital patient can be represented in a treatment plan, which has both internal coordination relationships (some tests must be done in a particular order or within certain time limits) and external coordination 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 on-line 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 at different locations nearly simultaneously 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.

All of these types of scenarios have been typically approached by building systems where complex agents have an internal representation of their own plans (and how they relate to the plans of other agents). Examples include CSC agents [9], or unrolling each agent's view of the HTN into a Markov decision process over which MDP techniques can be applied [10], or translating it into a Simple Temporal Network and applying STN techniques [23].

We take a radically different approach. Rather than putting the HTN inside of complex agents, one might put stigmergic agents inside of the HTN. Coordination would be achieved, not by conventional inter-agent 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.

This paper demonstrates this approach by showing how stigmergy can operate on a HTN. Specifically, we work with a

Cite as: Stigmergic Reasoning over Hierarchical Task Networks, Parunak et al., *Proc. of 8th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2009)*, Decker, Sichman, Sierra, and Castelfranchi (eds.), May, 10–15, 2009, Budapest, Hungary, pp. XXX-XXX. Copyright © 2009, International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

dialect of the TÆMS task language [8] that emphasizes the importance of resources, both real and virtual, in coordination (thus resource-TÆMS or rTÆMS). Section 2 reviews TÆMS and the rTÆMS dialect. Section 3 shows how one can apply stigmergy to an rTÆMS graph (along the way analyzing some “obvious” approaches that do not work). Section 4 reports on experiments that demonstrate the effectiveness of 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, together with two kinds of relations among them: a hierarchical structure relating tasks to their subtasks, and other relations constraining the order of execution among the tasks. For the sake of concreteness, we focus on TÆMS as a specific instance of an HTN formalism [1, 8], and argue for the importance of emphasizing resources, leading to the resource TÆMS (rTÆMS) dialect.

2.1 Introduction to TÆMS

To introduce the basic features of TÆMS, Figure 1 uses it to represent the dining philosophers problem.

- Ovals (“Argue,” “Think₁”) are *tasks* and *subtasks*. Tasks and subtasks may be associated with one or many agents, and can be subdivided into lower-level activities.
- 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*, which capture precedence constraints.
- Inverted triangles (“Forks”) are *resources*, which are produced and consumed by methods.

The dashed lines show the subgraphs accessible to each agent, known as their *subjective* graphs. The overall graph, which describes the complete problem, is the *objective* graph.

As methods execute successfully, they produce quality that flows up to their dominant subtasks and tasks. Each task or subtask has a Quality Accumulation Function (QAF) that describes how quality from its subordinates is combined. QAFs provide a much more nuanced way to capture what other HTN’s represent as AND and OR branches. For example, a Min QAF asserts that the quality of a task is the minimum of the incoming quality levels, and thus requires all subtasks or methods to execute before the task receives any quality (an AND), while a Max QAF corresponds to an OR, yielding nonzero task quality as soon as any subtask has executed. More complex functions are also possible.

The version of TÆMS in [8] explicitly represents dependencies between events and resources. C_TÆMS [1] implicitly acknowledges resources, but does not attempt to represent them.

Sometimes we need a more general vocabulary than the canonical TÆMS terms. Tasks, subtasks, methods, and resources are all *nodes* in a graph, whose *relations* are provided by non-local effects, resource dependencies, and QAFs. Tasks, subtasks, and methods all describe *events*.

Because HTN’s are graphs, and because graphs are a convenient formalism for stigmergic

environments, we begin by inquiring whether we can use an HTN directly as an environment for stigmergic interaction. Naively, we might imagine that the events in an HTN could serve as the places of our environment in which agents deposit digital pheromones. On reflection, this approach poses a problem. Commonly, a single agent is responsible for each event in an HTN. A task may have subtasks executed by different agents, but the atomic events (the methods) are the responsibility of single agents, and responsibility for the supertask may also be assigned to a single agent (the “manager”).

However, even private events must access shared resources, as Figure 1 illustrates. So it is natural to try to use resources as the basis for coordination. Competition for resources can be viewed as a prototypical form of stigmergic coordination [17], suggesting that resources are natural candidates for the places of a stigmergic environment. In fact, the Dining Philosopher’s problem is an example of a problem whose analysis must include shared resources.

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 [8], resources are only involved in some task interrelationships. Subtasking relations and non-local effects are described without any reference to resources, and [1] finds it possible to ignore them altogether.

This section argues that in fact resources (or things that behave like resources) are ubiquitous in TÆMS structures, and lie behind both subtasking and non-local effects relations. In fact, we argue that a fully elaborated TÆMS structure is a bipartite graph, whose two components are events and resources. Central to this vision is the notion that what TÆMS calls “quality” has similar functions to resources, and can be treated as a resource.

2.2.1 Quality as a Resource

The notion of “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 a number of operations add successive features to parts that move between them. For instance, 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.

If we were in fact modeling a manufacturing system, the parts that move from one operation to the other would naturally be 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. We ought to be able to 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 QAFs. Space limitations require this discussion to be abbreviated. A complete description of this mapping is available on-line in a technical report [14].

2.2.2 Non-Local Effects (NLE)

C-TÆMS defines four NLE relations: *Enables*, *Facilitates*,

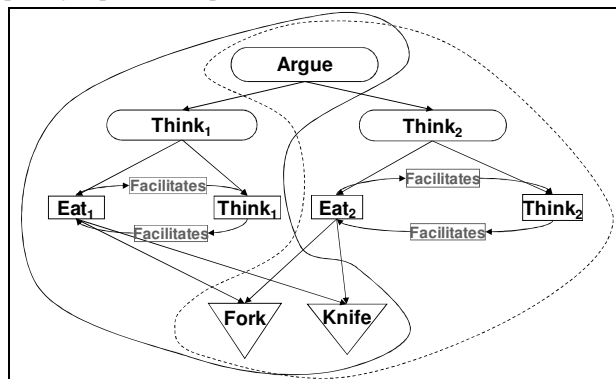


Figure 1: Agent Interaction through Resources.--Both agents must access a knife and a fork in order to eat.

Disables, and *Hinders*. All of these have 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.

To accommodate the delay, 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,¹ produces the same amount of quality as its input, and makes it available to its target. Like other methods, a NLE method requires some computational element to execute it. The natural candidate is the environment [24].

Figure 2 illustrates this mapping for the *Enables* relation. We use *Limits* rather than *Consumes* to join the quality resource to the *EnableMethod* so that multiple NLE's can draw on the same quality without interfering with one another.

For *Enables*, both *Limits* relations require their source resource to be above 0. The same diagram works for *Disables* as well, but now *Limits2* requires its source resource *not* to exceed 0.

If the Delay in the *Enables* or *Disables* is 0, the *EnableMethod* can be eliminated, and T1's quality can directly Limit T2.

Facilitates and *Hinders* compute a multiplier for the cost, quality, and duration of the target. Instead of simply copying the source's quality to its output, it now produces the three multipliers and makes them available to its target. *Facilitates* and *Hinders* use exactly the same structure, and differ only in how they compute the multipliers [14].

2.2.3 QAFs

The dictionary of QAFs [8] includes some (e.g., *SeqMax*, *SeqMin*) that involve sequencing constraints among subtasks. These have disappeared in [1], and in fact are syntactic sugar that can always be handled using the apparatus of NLE's outlined above. The QAFs that remain are *Sum*, *Max*, *Min*, *SyncSum*, *SumAnd*, and *ExactlyOne*. We call the first three "simple QAFs" and the last three "complex QAFs."

The basic idiom for handling subtasks and QAFs is that subtasks produce their individual qualities, and (in the case of complex QAFs) additional resources. The supertask then combines the individual qualities into its quality. Figure 3 shows the resulting structure. In each case, whenever the Task wishes to compute its quality, it simply applies the appropriate function (*Max*, *Min*, *Sum*) across the qualities of its subtasks.

SumAnd and *ExactlyOne* can be modeled by having each subtask produce an additional resource, a shared counter that indicates how many subtasks have executed. For *SumAnd*, the supertask adds up the qualities of the subtasks only if the counter indicates that all subtasks have executed. *ExactlyOne* adds them only if the counter is identically 1.

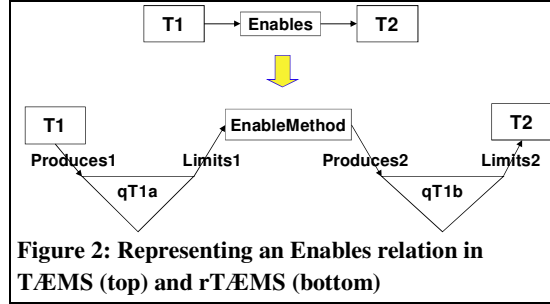


Figure 2: Representing an Enables relation in TÆMS (top) and rTÆMS (bottom)

and the other its quality. The supertask then selects the subtask(s) with the earliest start time and sums their qualities.

3. STIGMERGY OVER rTÆMS

It was surprisingly difficult to get stigmergy to work on a HTN. This section summarizes stigmergy, analyzes two approaches that did not succeed, then describes one that does.

3.1 Basic Stigmergic Approach

Our approach is based on polyagents [15], 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.

3.2 The Resource Dual of a TÆMS Graph

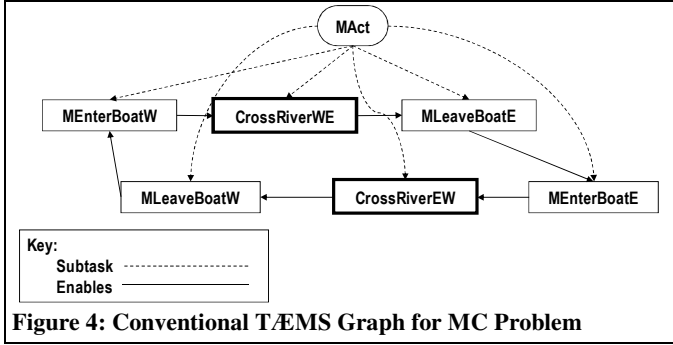
Section 2.1 argued that a network of tasks and methods is inadequate as a stigmergic environment because at least the methods, and often the tasks, are associated with single agents. Resources, not tasks, are the locus of most coordination. By considering quality as a virtual resource, we can represent any TÆMS graph as a bipartite graph whose nodes are resources and events.

Just as C-TÆMS constructs TÆMS graphs with no resources, it is possible to construct the resource dual of a TÆMS graph, containing only resources. For example, Figure 4 exhibits a resource-free TÆMS graph for a Missionary in the Missionary-Cannibal problem, in the spirit of [1]. 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 [14] provides the details of elaborating this structure into a full rTÆMS graph, then deriving the resource dual, a hypergraph of resources with a recurrent core (with neither sinks nor sources) shown in Figure 5.

Note the following interesting features of this structure.

- The outer ring of qXXXX resources captures the same structure as the ring of events in Figure 4, so we have not lost any structure with respect to that figure. In general, following paths among quality resources will reproduce the same paths as *Enables*, *Disables*, *Facilitates*, and *Hinders* relations.

¹ This is the DTT specification favored by Decker [4]. An alternative specification permits quality to accumulate as a method executes.



- The structure among the population resources (mX, cX) and between them and the quality resources capture important constraints on what events should be allowed or disallowed.
- Only two of the six nodes in the recurrent core of Figure 4 are shared with the TÆMS graphs for other agents, affording minimal opportunity for stigmergic interaction among agents. In contrast, eight out of twelve of the nodes in Figure 5 are shared.
- Observe not only the proportion of shared nodes, but also their connectivity. Figure 4 has pairs of private methods (MLeaveBoatW and MEnterBoatW, MLeaveBoatE and MEnterBoatE) whose transitions are not constrained by any public node. Thus the graph does not provide the structure needed to propagate influences from one agent to another to guide transitions between these activities. Figure 5 has no such pairs. Paths of public nodes (with bold outlines) connect every pair of private nodes, providing information 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 5. The traces of our experiments showed that by omitting events from the agents' environment, we lost important distinctions among different ways that the resources could be replenished or consumed, leaving the ghosts with too little information to solve the problem. An event hierarchy without resources (a C-TÆMS graph) offers no suitable places for stigmergic interaction, but the task dependencies it encodes are critical to guide that interaction.

3.3 Full rTÆMS Graph

Our next effort placed the ghosts on a TÆMS graph fully elaborated as a bipartite graph of resources and events. Ghosts deposit and sense pheromones on both resources and events, but the two have different functions. Pheromone on a *resource* enables the methods that require that resource. It is impossible to execute a method without the required resources. 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.

To support these semantics, ghosts deposit pheromone on resources and methods at different times.

- As ghosts move outward away from their avatar, exploring alternative futures, they deposit pheromone on the resources that 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 hier-

rarchy in order to move from one method to another. In the process, they bottlenecked at branch points in the hierarchy, and made little progress toward completing the task.

The hierarchical structure of a HTN shows us the logical structure of how methods and subtasks are organized into tasks, but does not highlight the causal flow of a process. In a hierarchy, node proximity represents the logical structure of the task, but 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. Each method is in fact part of a hierarchical task structure, but the entity's movement is from method to method, not from method to subtask to task and back down. The bottlenecking we experienced 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.

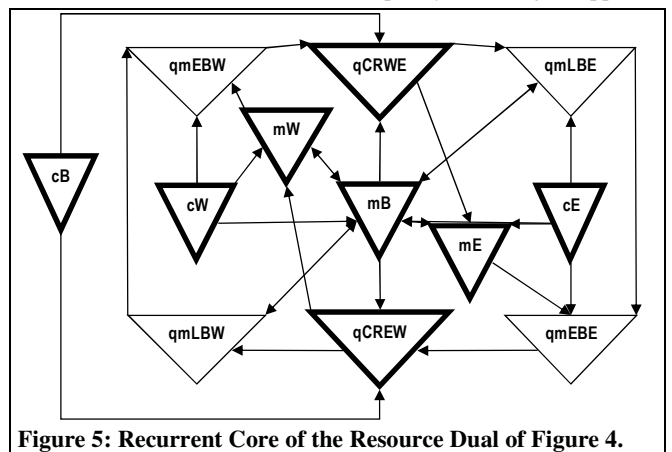
3.4.1 The Decomposed rTÆMS Graph

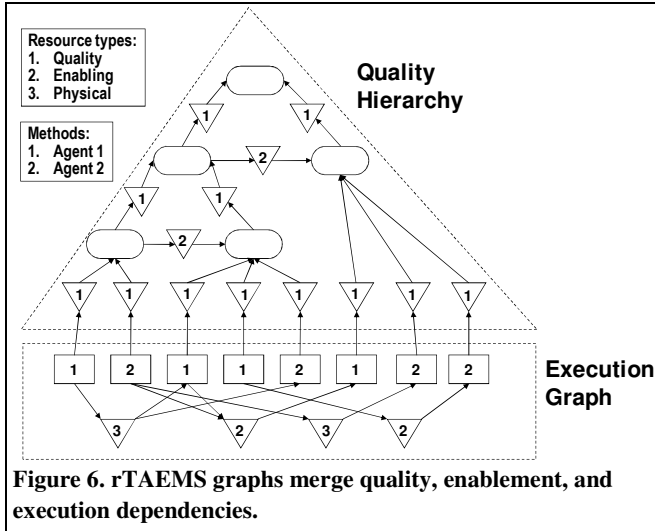
Any fully elaborated rTÆMS graph can be rearranged into an execution graph and a quality hierarchy (Figure 6).

The methods and the resources (physical or virtual) that constrain their sequencing form the *Execution Graph*. Our agents (avatars and ghosts) live on the execution graph, not on the rTÆMS hierarchy. For clarity, the execution graph in the figure is incomplete, because it does not include 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 will instantiate 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.

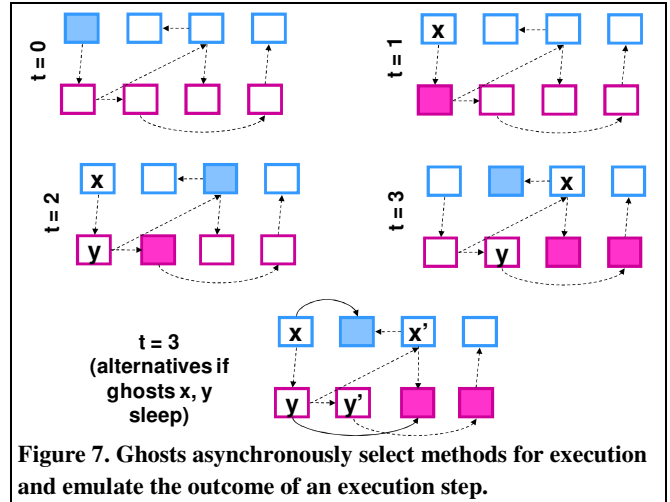
The other function uses the current state of quality to propagate method desirability back down the graph. The desirability of a method depends on its Quality Improvement Potential (QuIP), how much difference to the overall mission the quality it would produce would yield. For example, if two methods are OR'd into a subtask, the more quality has already accumulated at the subtask, the less of a 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: localization and activity. The environment *localizes* agents and their effects to reduce their computational burden. In addition, it is *active*, of-flooding 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

The function of an entity's polyagent is to find an optimal path through the Execution Graph. If multiple entities execute concurrently in the same Execution Graph, the stigmergic interaction between them by way of 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 is motivated by the scenario in Figure 7. Each frame shows a set of eight methods, four eligible for execution by avatar X (top row of each frame) and four eligible for execution by avatar Y (bottom row). The letters x and y indicate the current locations of one ghost from each avatar. The arrows indicate precedence relations among methods (omitting the intervening physical and virtual resources for clarity). Shaded squares are methods whose prerequisites are satisfied.

The first four frames indicate the outcome of a greedy policy, in which each ghost always picks a feasible method as soon as possible. However, another sequence (the fifth frame) might be possible, due to actions by other ghosts. In this case, if a ghost slept for a while, other ghosts might enable further methods that it could consider. To ensure that the search space includes these options, each ghost flips a weighted coin when it is activated to decide whether to go back to sleep or to take action.

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 two values: 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 execution duration of the node.

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 topologically 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 the performance of our method on the rTÆMS graph of Figure 8. 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 sub-task structure represented by the Quality Hierarchy. For clarity, we omit the resources that in fact occupy each link.

We demonstrate the behavior of our algorithm by comparing it to two simpler algorithms, on the graph of Figure 8 and two simpler, derivative graphs. Thus we execute $3 \times 3 = 9$ experimental configurations, each one replicated 25 times.

The three algorithms are:

A1. A random baseline, which selects methods at random without regard for either their enablement in the Execution Graph or their contribution of quality to the 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 [19], 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.

A2. A version of our algorithm that ignores QuIP information and selects methods based only on enablement and desirability. The effect is an algorithm similar to that described in [2].

A3. Our full algorithm, with selection among enabled methods determined entirely by 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 in our experiments: how rapidly the algorithm accumulates quality, and the variation among different runs.

Figure 9 shows plots of quality vs. time, with error bars indicating variability, for each of our nine configurations. The maximum quality available is 16 units (one for each method), except in N3, where the use of *Min* QAFs reduces it to 8. A2 and A3 reach maximum quality in 16 steps. A1 does so only in N1. Its average

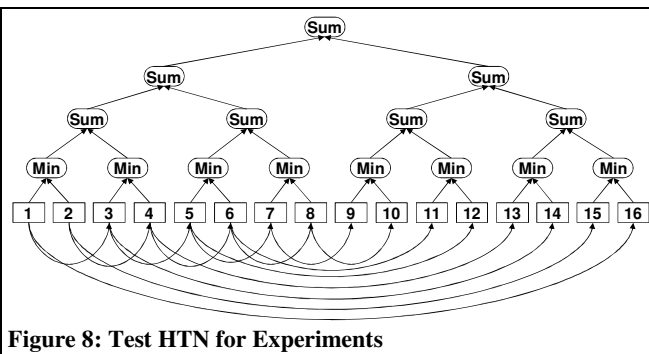


Figure 8: Test HTN for Experiments

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

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. Recall that 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

Based on the results obtained so far, we are planning to extend this work in a number of ways.

Our current experiments focus on a single avatar, so that we can clearly monitor the development and effect of the pheromone fields. To achieve the full promise of coordination among multiple entities, we will explore the interaction of 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, and conservation of physical resources.

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 advantag-

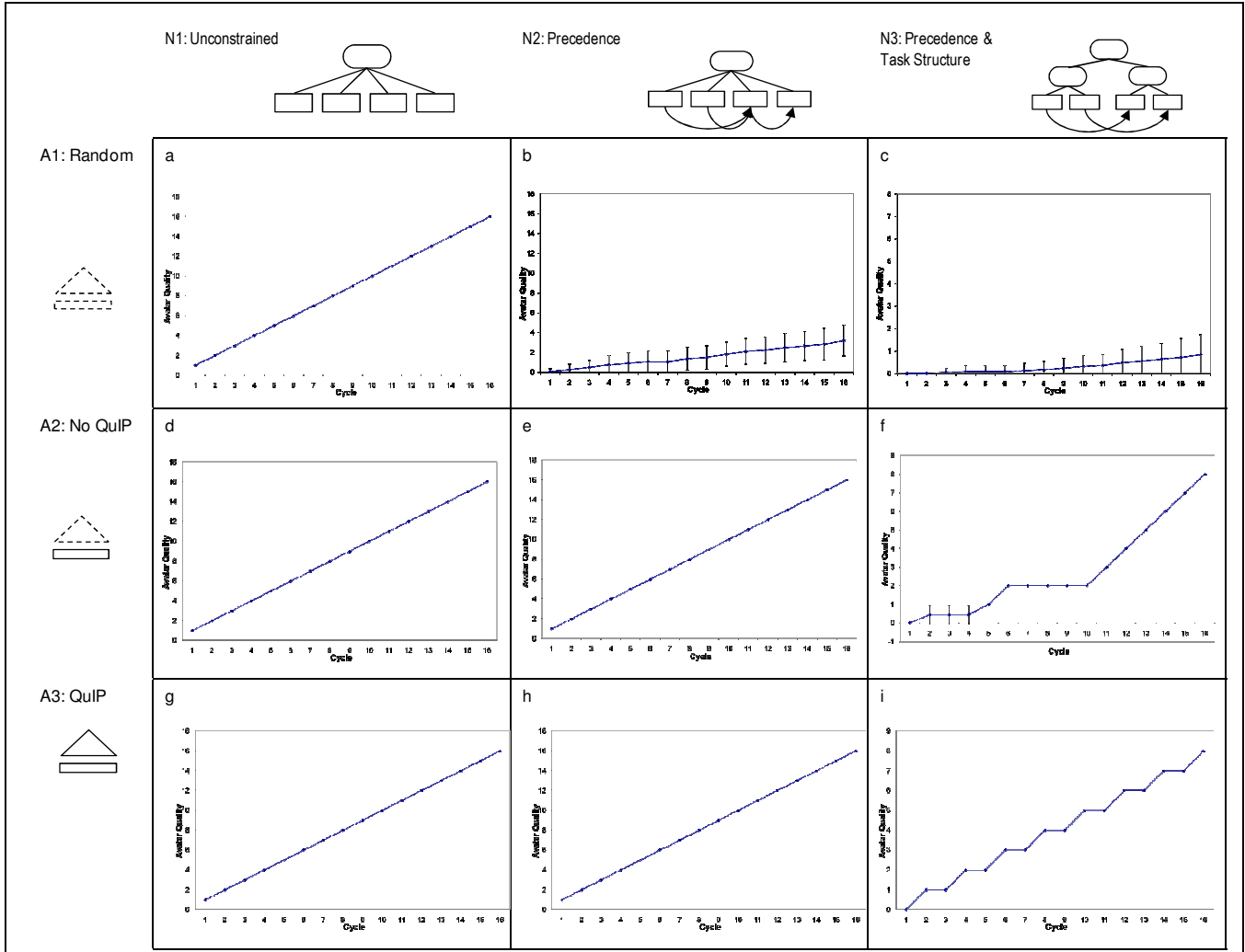


Figure 9: Experimental Results. Abcissa is cycle #, ordinate is avatar quality. Max quality is 16 in left and center columns, 8 at right.

es 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 QAFs) 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 in the context of HTN's has led to several important lessons.

- A semantic representation must contain locations that are shared among multiple agents in order to support stigmergic interaction. 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 must be made available to the agents.
- 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.

- An important criterion for the parts of the environment that localize the agents is that the topological neighborhood of the agents 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. ACKNOWLEDGMENTS

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. The authors are grateful to Prof. Keith Decker for extensive discussions and suggestions on this research.

8. REFERENCES

- [1] M. Boddy, B. Horling, J. Phelps, R. P. Goldman, R. Vincent, A. C. Long, B. Kohout, and R. Maheswaran. C_TAEMS

- Language Specification, Version 2.02. DARPA, Arlington, VA, 2006.
- [2] S. Brueckner. *Return from the Ant: Synthetic Ecosystems for Manufacturing Control*. Dr.rer.nat. Thesis at Humboldt University Berlin, Department of Computer Science, 2000. <http://dochoost.rz.hu-berlin.de/dissertationen/brueckner-sven-2000-06-21/PDF/Brueckner.pdf>.
- [3] W. Chen and K. Decker. The Analysis of Coordination in an Information System Application--Emergency Medical Services. In P. Bresciani, P. Giorgini, B. Henderson-Sellers, and M. Winiko, Editors, *Agent-Oriented Information Systems*, vol. 3508, *LNAI*, pages 36-51. Springer, New York, NY, 2005.
- [4] K. Decker. *Environment Centered Analysis and Design of Coordination Mechanisms*. Thesis at University of Massachusetts, Department of Computer Science, 1995. <http://www.cis.udel.edu/~decker/pubs/decker-thesis-tr.pdf>.
- [5] K. Decker and J. Li. Coordinating mutually exclusive resources using GPGP. *Journal of Autonomous Agents and Multi-Agent Systems*, 3(2):133-158, 2000.
- [6] M. Dorigo and T. Stuetzle. *Ant Colony Optimization*. Cambridge, MA, MIT Press, 2004.
- [7] P.-P. Grassé. La Reconstruction du nid et les Coordinations Inter-Individuelles chez *Bellicositermes Natalensis et Cubitermes sp.* La théorie de la Stigmergie: Essai d'interprétation du Comportement des Termites Constructeurs. *Insectes Sociaux*, 6:41-84, 1959.
- [8] B. Horling, V. Lesser, R. Vincent, T. Wagner, A. Raja, S. Zhang, K. Decker, and A. Garvey. The Taems White Paper. 2004. Web site, <http://dis.cs.umass.edu/research/taems/white/>.
- [9] R. T. Maheswaran, P. Szekely, M. Becker, S. Fitzpatrick, G. Gati, J. Jin, R. Neches, N. Noori, C. Rogers, R. Sanchez, K. Smyth, and C. Vanbuskirk. Predictability & Criticality Metrics for Coordination in Complex Environments. In *Proceedings of Seventh International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2008)*, Estoril, PT, 2008. <http://coordination.isi.edu/home/wp-content/uploads/2008/03/aamas2008paper.pdf>.
- [10] D. J. Musliner, E. H. Durfee, JianhuiWu, D. A. Dolgov, R. P. Goldman, and M. S. Boddy. Coordinated Plan Management Using Multiagent MDPs. In *Proceedings of AAAI Spring Symposium on Distributed Plan and Schedule Management*, Palo Alto, CA, 2006. <http://www.cs.umd.edu/users/musliner/papers/ss06-iu.pdf>.
- [11] H. V. D. Parunak. 'Go to the Ant': Engineering Principles from Natural Agent Systems. *Annals of Operations Research*, 75:69-101, 1997. <http://www.newvectors.net/staff/parunakv/gotoant.pdf>.
- [12] H. V. D. Parunak. A Survey of Environments and Mechanisms for Human-Human Stigmergy. In D. Weyns, F. Michel, and H. V. D. Parunak, Editors, *Proceedings of E4MAS 2005*, vol. LNAI 3830, *Lecture Notes on AI*, pages 163-186. Springer, 2006. www.newvectors.net/staff/parunakv/HumanHumanStigmergy2005.pdf.
- [13] H. V. D. Parunak. Real-Time Agent Characterization and Prediction. In *Proceedings of International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS'07), Industrial Track*, Honolulu, Hawaii, pages 1421-1428, ACM, 2007. <http://www.newvectors.net/staff/parunakv/AAMAS07Fitting.pdf>.
- [14] H. V. D. Parunak. HTN-Induced Stigmergic Environments. NewVectors division of TTGSI, Ann Arbor, MI, 2008. http://www.newvectors.net/staff/parunakv/rTAEMS_TechMemo.pdf.
- [15] H. V. D. Parunak and S. Brueckner. Concurrent Modeling of Alternative Worlds with Polyagents. In *Proceedings of the Seventh International Workshop on Multi-Agent-Based Simulation (MABS06, at AAMAS06)*, Hakodate, Japan, Springer, 2006. <http://www.newvectors.net/staff/parunakv/MABS06Polyagents.pdf>.
- [16] H. V. D. Parunak, S. Brueckner, M. Fleischer, and J. Odell. A Design Taxonomy of Multi-Agent Interactions. In *Proceedings of Agent-Oriented Software Engineering IV*, Melbourne, AU, pages 123-137, Springer, 2003. www.newvectors.net/staff/parunakv/cox.pdf.
- [17] H. V. D. Parunak, S. Brueckner, M. Fleischer, and J. Odell. A Preliminary Taxonomy of Multi-Agent Activity. In *Proceedings of Autonomous Agents and Multi-Agent Systems (AAMAS 2003)*, Melbourne, Australia, pages 1090-1091, ACM, 2003. www.newvectors.net/staff/parunakv/COX03.pdf.
- [18] H. V. D. Parunak, P. E. Nielsen, S. Brueckner, and R. Alonso. Hybrid Multi-Agent Systems. In *Proceedings of Proceedings of the Fourth International Workshop on Engineering Self-Organizing Systems (ESOA'06)*, Hakodate, Japan, Springer, 2006. <http://www.newvectors.net/staff/parunakv/ESOA06Hybrid.pdf>.
- [19] D. d. S. Price. A General Theory of Bibliometric and Other Cumulative Advantage Processes. *Journal of the American Society for Information Science*, 27(5-6):292-306, 1976. <http://garfield.library.upenn.edu/price/pricetheory1976.pdf>.
- [20] J. A. Sauter, R. Matthews, H. V. D. Parunak, and S. A. Brueckner. Demonstration of Digital Pheromone Swarming Control of Multiple Unmanned Air Vehicles. In *Proceedings of AAAI Infotech@Aerospace*, Arlington, VA, AIAA, 2005. <http://www.newvectors.net/staff/parunakv/aiaa05.pdf>.
- [21] R. Schoonderwoerd, O. Holland, and J. Bruten. Ant-like agents for load balancing in telecommunications networks. In *Proceedings of Autonomous Agents 97*, Marina del Rey, CA, pages 209-216, Association for Computing Machinery, 1997.
- [22] H. A. Simon. *The Sciences of the Artificial*. Cambridge, MA, MIT Press, 1969.
- [23] S. F. Smith, A. Gallagher, T. Zimmerman, L. Barbulescu, and Z. Rubinstein. Distributed Management of Flexible Times Schedules. In *Proceedings of Sixth International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2007)*, Honolulu, Hawaii, pages 472-479, 2007.
- [24] D. Weyns, H. V. D. Parunak, F. Michel, T. Holvoet, and J. Ferber. Multiagent Systems, State-of-the-Art and Research Challenges. In *Proceedings of Workshop on Environments for Multi-Agent Systems (E4MAS 2004)*, New York, NY, pages 1-47, Springer, 2004. http://www.cs.kuleuven.ac.be/~danny/e4mas_survey_2004.pdf.

