Stigmergic Reasoning over Hierarchical Task Networks (Extended Abstract)

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, usually used on simple problems, can be applied to more complex ones by encoding them in the agents' environment. We show how stigmergic agents can plan over a hierarchical task network, a resource-oriented dialect of the TÆMS language. 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, Agents, Interaction

1. INTRODUCTION

Most applications of stigmergy solve problems with simple semantics (e.g., routing). This constraint says more about the environment than it does about stigmergic interaction. Stigmergy transfers cognition from the agents to the environment, and can coordinate agent behaviors on a structure with complex semantics. A cognitively rich environment can yield cognitively complex outcomes among relatively simple agents [6].

One domain that seems to require higher levels of cognition is coordination in executing complex tasks. The task structure can be represented as a hierarchical task network (HTN). Planning and scheduling are usually approached by complex agents with an internal representation of their own plans (and how they relate to those of other agents). We take a different approach. Rather than putting the HTN inside complex agents, we put stigmergic agents inside the HTN. Coordination emerges, not from dialogs based on each agent's individual analysis of the HTN, but from interactions among the agents mediated by the structure of the HTN.

This paper demonstrates this approach by showing how stigmergy can operate on a HTN. Specifically, we work with a dialect of the TÆMS task language [3] 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, and summarizes our results.

Cite as: Stigmergic Reasoning over Hierarchical Task Networks, (Extended Abstract), H. V. Parunak, T. Belding, R. Bisson, S. Brueckner, E. Downs, R. Hilscher, *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. 1195–1196

2. TÆMS AND rTÆMS

An HTN is a collection of events, and two kinds of relations among them: a hierarchy relating tasks to subtasks, and other relations constraining the order of execution among tasks. For concreteness, we focus on the TÆMS HTN formalism [1, 3].

TÆMS primitives include decomposable tasks, atomic methods, non-local effects between tasks and methods, and resources, which are produced and consumed by methods. As methods execute, they produce quality that flows up to the tasks that they compose. Each task or subtask has a Quality Accumulation Function (QAF) that describes how quality from its subordinates is combined, providing a more nuanced way to capture what other HTN's represent as AND and OR branches.

Graphs are a convenient formalism for stigmergic environments. Can we can use an HTN directly as a stigmergic environment. Naively, we might use the events in an HTN 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.

However, even private events must access shared resources. So it is natural to try to use resources as the basis for coordination. Competition for resources is a form of stigmergy [5], so resources are natural candidates for the places of a stigmergic environment.

The full paper[1] defines "resource TÆMS" (rTÆMS). A fully elaborated rTÆMS graph is bipartite. Its two components are events (tasks and methods) and resources (which subsume not only physical resources but also quality and non-local effects).

3. STIGMERGY OVER rTÆMS

Polyagents [4] represent each domain entity (for example, each actor coordinating over the HTN) as a persistent *avatar* that sends out a continuous stream of apoptotic *ghost agents*. The ghosts interact through stigmergy, depositing and sensing digital pheromones in the network as they explore it. In the process, they explore a very large space of possible futures. The avatar chooses actions based on their survey. The ghosts reason through time using a book of pheromones, a set of snapshots of the environment and its pheromone deposits at each time step into the future, up to the limit of our exploration.

Getting stigmergy to work on a HTN was elusive. The full paper describes two unsuccessful approaches. We succeeded by

[1] http://www.newvectors.net/staff/parunakv/AAMAS09rTAEMS.pdf

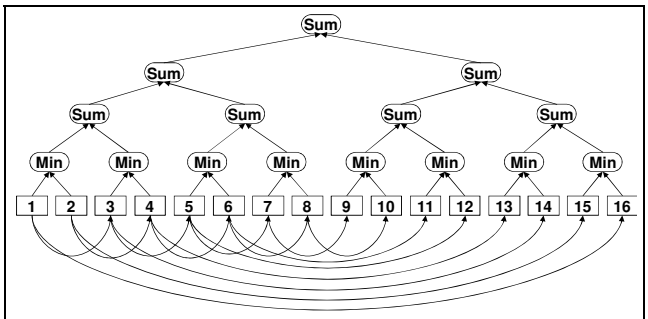

Figure 1: Test HTN for Experiments

decomposing the graph into two parts. The methods and the resources (physical or virtual) that constrain their sequencing form the *Execution Graph*. The subtask hierarchy computes and communicates the quality achieved by the system, so we call it the *Quality Hierarchy*. It keeps each method informed of 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 MAX'd into a subtask, as quality from one accumulates at the parent, the other becomes less important.

The distinction between the Execution Graph and the Quality Hierarchy highlights two functions of a stigmergic environment: it *localizes* agents to reduce their computational scope, and it is *active*, offloading some computation from the agents. The Execution Graph localizes by restricting an agent's neighborhood to the most relevant methods for it to consider, while the Quality Hierarchy does quality estimation in addition to the usual actions of pheromone aggregation, evaporation, and propagation.

Three population of agents swarm over this structure.

Each *actor* coordinating over the graph maintains a swarm of Entity Ghosts that move spatio-temporally over the Execution Graph. Their choice of which method to execute when depends on the availability of input resources, the capacity of output resources, and the method's current QuIP. These ghosts deposit pheromones in the methods to indicate the likelihood that a given method will be executed at a given time.

Each *resource* has a swarm of Resource Ghosts that move temporally through the book of pheromones to estimate the likely level of the resource at each future time step, based on the execution estimates provided by Entity Ghost pheromones on methods.

Each *node in the Quality Hierarchy* has a swarm of Quality Ghosts that move temporally through the book of pheromones to estimate the node's likely quality level as a function of time, analogous to the operation of the Resource Ghosts. These levels provide feedback for the QuIP of each method over time.

We demonstrate our method on Figure 1, which captures two kinds of constraints in a HTN: precedence constraints among methods (in the Execution Graph), and the subtask structure (in the Quality Hierarchy). For clarity, we omit the resources that occupy each link. We compare our algorithm to two others, on Figure 1 and two derivative graphs. 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 in the Quality Hierarchy.

A2. A version of our algorithm that ignores QuIP and selects methods based only on enablement and desirability, similar to [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 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 runs. All three algorithms perform the same on N1. On N2, A2 and A3 are equal and dominate A1. On N3, A3 dominates A2, which dominates A1.

This result highlights two kinds of complexity in an HTN, one due to precedence constraints, the other to the task structure through which methods deliver quality. One can locate HTN's in this 2-D space, and distinguish solution methods (classical as well as swarming) by the dimension(s) they address.

Our research[2] leads to several important lessons.

- A representation must contain locations that are shared among multiple agents in order to support stigmergic interaction, leading to the rTÆMS 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 same agents swarm over all node types is not the best way to give them this information. Some parts of a graphical representation of a domain may localize the agents, while other parts can support environmental actions.
- 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.

4. 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*. Thesis at Humboldt University Berlin, Department of Computer Science, 2000.

[3] B. Horling, V. Lesser, R. Vincent, T. Wagner, A. Raja, S. Zhang, K. Decker, and A. Garvey. The Taems White Paper. Multi-Agent Systems Lab, University of Massachusetts, Amherst, MA, 2004. http://dis.cs.umass.edu/research/taems/white/.

[4] 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)*, Springer, 2006.

[5] 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)*, pages 1090-1091, ACM, 2003.

[6] H. A. Simon. *The Sciences of the Artificial*. Cambridge, MA, MIT Press, 1969.

[2] Conducted with the support of the office of Naval Research (Contract # N00014-06-1-0467). The results do not necessarily reflect the opinion of the sponsor. The authors thank Prof. Keith Decker for extensive discussions and suggestions.