HIVE-BDI: EXTENDING JASON WITH SHARED BELIEFS AND STIGMERGY

Matteo Barbieri and Viviana Mascardi
Department of Computer Science (DISI), University of Genova, Genova, Italy

Keywords: BDI agents, MASs, Stigmergy, Belief sharing, Agent coordination.

Abstract: The classic BDI model focuses on the internal functioning of a single-agent architecture. Neither shared beliefs nor spontaneous and indirect coordination via the environment are supported. We describe *Hive-BDI*, an extension of the Jason BDI-style language with shared beliefs implemented via the logic-based coordination language ReSpecT and with stigmergy obtained via digital pheromones. A case study where robots roaming an unknown environment collaborate for creating a map demonstrates the feasibility of our approach.

1 INTRODUCTION

The classic BDI model (Rao and Georgeff, 1991) focuses on the internal functioning of a single-agent architecture: the way events taking place in the environment are perceived and the way agents can organize themselves by sharing mental attitudes are not addressed. No form of spontaneous and indirect coordination via the environment (stigmergy (Bonabeau, 1999)) is supported, and information belonging to an agent's private set of beliefs can be shared only by explicit communication. Many extensions of this basic model to cope with groups, teams and organizations formation have been proposed in the literature (Hepple et al., 2008), and the research interest in achieving agent coordination via stigmergy has increased in the last ten years (Valckenaers et al., 2001; Parunak et al., 2002; Ricci et al., 2007).

The usefulness of a BDI framework supporting both belief sharing and stigmergy is easy to understand. Imagine a simple scenario where several robots governed by rational agents are roaming an unknown environment, collecting data as they move and processing them in order to create a shared map. In the classic BDI model, a previously designated coordinator agent should act as a "shared blackboard" so that all information collected by every agent in the system can coexist in the belief base of a single agent. If an agent needed information about an area inspected by another agent, he should query the coordinator. Also, two or more agents that wanted to coordinate in order to define a joint strategy should adopt sophisticated negotiation protocols mediated by the coordinator.

Instead of an adaptation of the existing BDI model, a new model where each agent, still retaining his personal belief base, can also access a shared knowledge base might prove suitable for avoiding bottlenecks and for ensuring fault-tolerance. Shared information should be immediately available to every other agent in the MAS or in the group of registered agents and coordination should be achieved via the environment.

Despite of the advantages of beliefs sharing and stigmergy, to the best of our knowledge none among the most widespread BDI frameworks has been extended yet to support both features. To overcome this limitation we designed and implemented *Hive-BDI* that extends Jason (Bordini et al., 2007) with shared beliefs by taking advantage of the logic-based coordination language ReSpecT (Omicini and Denti, 2001), and with stigmergy by introducing digital pheromones.

The paper is organized in the following way: Section 2 describes the technologies upon which Hive-BDI grounds and analyses the related work; Section 3 gives hints on the Hive-BDI implementation and briefly discusses Hive-BDI at work; Section 4 concludes.

2 BACKGROUND AND RELATED WORK

Belief Sharing in BDI Settings. Some approaches and methodologies for belief sharing in MASs already exist: Glinton et al. (Glinton et al., 2010) present a formal model and methodology for analyzing and designing two types of multi-agent belief sharing models, based on Potts and Random Cluster Statistical Mechanics models respectively, and distinguished by the mechanism used for information exchange. Other works by the same research group address the belief sharing problem by taking a perspective of strategy optimization (Velagapudi et al., 2007; Velagapudi et al., 2008). However, very few attempts to address belief sharing in BDI languages and frameworks have been made. According to a recent paper by Hepple et al. (Hepple et al., 2008) that analyses the state-of-the-art in agent organization in BDI languages, only Pynadath et al.'s TEAMCORE (Pynadath et al., 2000) takes belief sharing as a characterizing feature for agent organizations ((Hepple et al., 2008), p. 82, Table 1). An approach similar to Pynadath's one is discussed in (Vieira et al., 2007) where, as examples of advanced communication features, Vieira et al. show plans that allow agents to reach shared beliefs and ensure that agents are kept informed of the adoption of their goals by other agents.

Stigmergy in Cognitive MASs. Stigmergy (Bonabeau, 1999) is a mechanism of spontaneous, indirect coordination between agents, where the trace left in the environment by an action stimulates the performance of a subsequent action, by the same or a different agent. Stigmergy can be integrated in a MAS as a different method of communication between agents, which exchange information by altering the environment releasing digital pheromones (Dorigo et al., 1996).

We are aware of only three works dealing with stigmergy in MASs consisting of cognitive agents. In (Ricci et al., 2007), the impact of stigmergy on the structure and organization of MASs based on cognitive agents is explored. Coffey and Clark present the structure of a hybrid architecture for robot control where a BDI-style planning layer manipulates a plan library in which plans are comprised of hierarchical, suspendable and recoverable teleo-reactive programs (Coffey and Clark, 2006). A shared belief store can be used by groups of agents. A foraging agent scenario where an ant-style pheromone-laying algorithm directs the agent's search by guiding it away from previously explored areas in a pseudo-random walk is used to demonstrate the approach. Because of the integration of shared beliefs and stigmergy, Coffey and Clark's work is very close to ours, although it uses a very different infrastructure. Finally, in (Piunti and Ricci, 2009) Piunti and Ricci investigate the use of cognitive artifacts in MAS, as computational entities designed to store, process and make available information relevant for agents to coordinate their cooperative distributed activities. The feasibility of their proposal has been experimented in Ja-Ca (Jason + CArtAgO, a platform for developing artifact-based environments for MAS (Ricci et al., 2008)). Although their work does not explicitly deal with stigmergy, it is worth mentioning since it might easily support it and because it is based on Jason, as ours.

Jason. Jason (Bordini et al., 2007) is a Java-based interpreter for an extended version of AgentSpeak (Rao, 1996) which allows a high level of customization. Our main extension to Jason affected the BeliefBase class - as far as belief sharing is concerned - and the Environment class - as far as the digital pheromones are concerned. The purpose of BeliefBase class is to manage the agent's beliefs. Three methods handle tuple addition, removal and lookup (checking whether a tuple exists in the belief base). The Environment class implements the simulated environment in which agents are situated. Its purpose is to update agents' percepts and react to their actions upon itself. The default Jason library includes two classes which may be used to implement an environment represented as a grid: GridWorldModel and GridWorldView.

ReSpecT. ReSpecT (Omicini and Denti, 2001) is a logic-based coordination language aimed at defining the behavior of tuple centers in order to coordinate autonomous computational entities, such as software agents. Written in Java, its package includes several components, in particular it provides Java classes for the creation and management of Tuple Centers, namely blackboards where agents can write data in the form of tuples. There are five main primitive operations which can be executed upon a ReSpecT Tuple Center (TC): *out(T)* writes a new tuple T on the TC; *rd(T)* retrieves a tuple X unifying with T from the TC (blocking); *in(T)* extracts a tuple X unifying with T thus deleting it from the TC (blocking); *rdp(T)* is the non-blocking variant of *rd(T)*; and *inp(T)* is the non-blocking variant of *in(T)*.

3 HIVE-BDI AT WORK

In this section, we provide a short overview of the Hive-BDI implementation and an example of use.

More details on both issues can be found in (Barbieri, 2010).

Shared Beliefs. The Hive-BDI shared knowledge base is implemented with a ReSpecT Tuple Center to which each agent's BeliefBase Class connects via a Socket. The shared knowledge base is composed of two parts: the Hive Belief Base (referred to as HiveBB in the sequel) and the Hive Server. Since the Hive-BDI model still retains personal agent's beliefs which are not accessible from the outside, each time an operation on agent's beliefs is invoked, it is necessary to distinguish whether the belief must be added, removed or searched for in the *shared* belief base or in the agent's *private* one. The distinction is possible thanks to the *mustshare* annotation that the Hive-BDI programmer must add to those agent's beliefs that will be stored in the shared belief base. The HiveBB is implemented as an abstract class that extends the DefaultBeliefBase Jason class. The Hive Server is a daemon which can run on a remote machine and whose purpose is storing shared beliefs and making them available to the agents in the system. After deciding that a belief must be shared, a connection with a remote Hive Server is established by the HiveBB class. Then all information required for the transaction is encapsulated in an object sent via a socket to the Hive Server where it is unpacked and the request is processed. Then another object is created and filled with response data (the result of requested operation and additional data.

An agent acts upon his belief base in three ways: adding, retrieving, deleting a belief to/from his knowledge base. Adding a new belief to the shared belief base is a little more complex than just writing a new tuple to the tuple center: this because in Jason's default implementation, if we want to add a belief to the Knowledge Base and another belief having the same functor, arity and arguments but different annotations[1] is already present in it, the new belief is not added: instead its annotations are merged with those of the belief already present. Retrieving a belief is actually quite a simple operation to implement, since the method to override does not really take care of selecting which belief to return: its task is just to narrow down the range of beliefs suitable for unification with the one passed as parameter. The default implementation coming with Jason selects beliefs having the same *functor* and *arity* as that one. Finally, deletion of a belief actually deletes the *source* of the belief itself from among its annotations. The belief is actually deleted when it has no more sources. The method responsible for belief deletion takes this behavior into account.

[1]Two beliefs having these characteristics will be called *similar*.

Digital Pheromones. In order to implement a pheromone-based stigmergy mechanism we created a new HouseWorldEnvironment class that includes properties *model* and *view* (whose classes are respectively HouseWorldModel and HouseWorldView, which, in turn, extend GridWorldModel and GridWorldView). We also defined a new entity *pheromones*. Since we need to store information about the "scent" and intensity of pheromones (i.e. *which* agent released it and the *intensity* of the trail), the HouseWorldModel class has a private property *pheromones* holding a bi-dimensional array of *Pheromone* objects. To make an example, if a cell of coordinates $(3,7)$ contains pheromones, all other information (intensity and scent, in this case) can be extracted from the object in *pheromones[3][7]*. Each time an agent moves in the environment he releases pheromones in the cell where he ends his movement. The release of pheromones is implemented basically in two steps inside the *addPheromones* method in the *HouseWorldModel*: first a check is made to see whether the (x,y) cell where the agent currently stands is already marked as containing pheromones. Then a new *Pheromone* object is created and put in *pheromones[x][y]*, potentially replacing information about pheromones already present in that cell. The constructor of class *Pheromone* takes two arguments: a string which identifies the agent releasing pheromones, and the intensity of the trace left which determines how long the trace will remain on the cell before disappearing because of evaporation.

Hive-BDI at Work: Mapping an Unknown Environment. We validated the potential of a model such as Hive-BDI to implement a MAS where a few explorer agents coordinated by a leader named queen, coherently with the Hive metaphor, are able to build a map of an unknown environment. Explorer agents represent robots initially placed more or less randomly in the environment whilst the queen has no physical incarnation. Once the simulation starts, each explorer begins to map the room he is in, then moves to the next unexplored room and so on until the whole environment has been mapped. Each time an agent perceives pheromones belonging to someone else he starts following the trail until he meets the agent who released it: at this point they merge the information collected up to that moment, adopting a common reference system for coordinates and thus updating their beliefs. Figure 1 shows the graphical output of a simulation run. Blue circles are agents, green tiles represent pheromone marks released by agents. Different shades of green represent pheromones having different intensity. The experiments we carried out on this simple case study were encouraging and demonstrated the suitability of the Hive-BDI model and of its implementation to cope with distributed coordination problems of this kind.

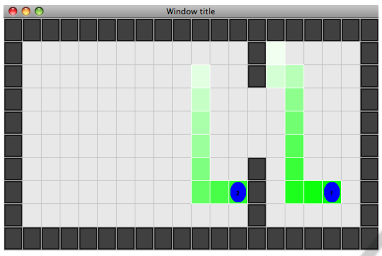

Figure 1: Screenshot taken from a simulation run.

4 CONCLUSIONS

Hive-BDI is suitable for modeling MASs composed by a relatively large number of agents with limited capabilities: the system draws its strength from the cooperation and coordination of agents, giving birth to an *emerging intelligence*.

Of course more complex agents, fully exploiting the BDI potential, can be part of the MAS: the result would be a MAS involving agents that, despite being homogeneous and fully interoperable, can be clustered in different layers according to their reactivity/deliberativity. More reactive layers would consist of many simple agents fully exploiting the Hive-BDI features whereas more deliberative layers would consist of "pure" Jason agents.

Using simpler entities means having less requirements in terms of computational and material resources. This may be a desirable feature in real scenarios where software agents control physical robots. When talking about robots, limited capabilities (e.g., limited strength or storage capacity, in the case of a *cleaning robot*) may imply limited *size*, thus allowing such machines to operate in environments otherwise precluded to bigger ones.

As far as the future of this research activity is concerned, we are working at enhancing the shared knowledge base class by fully exploiting the ReSpecT tuple center and we are experimenting Hive-BDI on other case studies.

To conclude, the possibility given by Hive-BDI to define *conditions* in an agent's plan's *context*, referring to the shared knowledge of the *hive cluster* rather than the agent's personal belief base really allows to *think* of the MAS as a single *entity*, and therefore to write plans accordingly: this actually would represent a *paradigm shift* in Multi Agent System programming.

REFERENCES

Barbieri, M. (2010). Hive-BDI extending BDI agents with shared memory. Bachelor's thesis, University of genova. http://www.disi.unige.it/person/MascardiV/Download/Barbieri-Matteo.pdf.

Bonabeau, E. (1999). Editor's introduction: Stigmergy. *Artificial Life*, 5(2):95–96.

Bordini, R. H., Wooldridge, M., and Hübner, J. F. (2007). *Programming Multi-Agent Systems in AgentSpeak using Jason*. John Wiley & Sons.

Coffey, S. and Clark, K. (2006). A hybrid, teleo-reactive architecture for robot control. In *MARS'06*.

Dorigo, M., Maniezzo, V., and Colorni, A. (1996). The Ant System: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics Part B: Cybernetics*, 26(1):29–41.

Glinton, R., Sycara, K., Scerri, D., and Scerri, P. (2010). The statistical mechanics of belief sharing in multi-agent systems. *Inf. Fusion*, 11(3):256–266.

Hepple, A., Dennis, L. A., and Fisher, M. (2008). A common basis for agent organisation in BDI languages. In *LADS 2007*.

Omicini, A. and Denti, E. (2001). Formal ReSpecT. *Electr. Notes Theor. Comput. Sci.*, 48.

Parunak, H. V. D., Brueckner, S., and Sauter, J. A. (2002). Digital pheromone mechanisms for coordination of unmanned vehicles. In *AAMAS 2002*. ACM.

Piunti, M. and Ricci, A. (2009). Cognitive use of artifacts: Exploiting relevant information residing in MAS environments. In *KRAMAS 2008*.

Pynadath, D. V., Tambe, M., Chauvat, N., and Cavedon, L. (2000). Toward team-oriented programming. In *ATAL'99*.

Rao, A. S. (1996). AgentSpeak(L): BDI agents speak out in a logical computable language. In *MAAMAW'96*.

Rao, A. S. and Georgeff, M. P. (1991). Modeling rational agents within a BDI-architecture. In *KR'91*. Morgan Kaufmann.

Ricci, A., Omicini, A., Viroli, M., Gardelli, L., and Oliva, E. (2007). Cognitive stigmergy: Towards a framework based on agents and artifacts. In *E4MAS 2006*.

Ricci, A., Viroli, M., and Omicini, A. (2008). The A&A programming model and technology for developing agent environments in MAS. In *ProMAS 2007*.

Valckenaers, P., Van Brussel, H., Kollingbaum, M. J., and Bochmann, O. (2001). Multi-agent coordination and control using stigmergy applied to manufacturing control. In *EASSS 2001*.

Velagapudi, P., Prokopyev, O., Scerri, P., and Sycara, K. (2008). A token-based approach to sharing beliefs in a large multiagent team. In *Optimization and Cooperative Control Strategies*.

Velagapudi, P., Prokopyev, O., Sycara, K., and Scerri, P. (2007). Maintaining shared belief in a large multiagent team. In *FUSION 2007*.

Vieira, R., Moreira, A. F., Wooldridge, M., and Bordini, R. H. (2007). On the formal semantics of speech-act based communication in an Agent-Oriented Programming language. *J. of Art. Intell. Research*, 29.