Finding Agreed Plans*

Gerhard Wickler
AIAI, University of Edinburgh
g.wickler@ed.ac.uk

Abstract

The aim of this paper is to describe and solve a new type of planning problem in which a group of peers need to find and agree upon a plan that solves a given problem. Classical planning only addresses the finding of a plan, whereas agent research defines the concept of commitment which we will use to define the "agreed" aspect of a plan. The problem will be solved by introducing a level of indirection.

Introduction

The classical planning problem [Ghallab *et al.*, 2004, section 2.3] has been extended in a number of ways. One of these extensions is the *distributed (multi-agent) planning problem* [Durfee, 1999]. In fact there is more than one problem that can be described by this term. In this paper, we will add yet another aspect to the problem by requiring that a plan which is a solution to the planning problem must be an *agreed* plan.

Notions of Distributed Planning

While the classical planning problem has a widely agreed definition, the distributed version can be defined in one of two fundamentally different ways. Firstly, it can be defined exactly as the classical planning problem, but the planning process that generates the solution plans is distributed over a number of computational resources. This could be to speed up the plan generation process, or it could be because the planning knowledge is distributed over the same resources. Secondly, it can be defined as an extension of the classical planning problem in which a solution plan must include an assignment of actions in the plan to agents executing these actions. This could be to speed up plan execution or because multiple agents have different capabilities that must be combined. The difference between the two approaches lies in what is distributed: the plan generation process or the plan execution.

*Sponsored by the European Research Office of the US Army under grant number N62558-06-P-0353. The authors' organizations and research sponsors are authorized to reproduce and distribute reprints and on-line copies for their purposes notwithstanding any copyright annotation hereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of other parties.

The Collaborative Multi-Peer Planning Problem

A third variant would of course be to have both distributed amongst a group of agents, and this is the version we have chosen to adopt for this paper: a group of agents has a set of goals for which they must find a plan and which assigns actions in the plan to agents for execution. The initial state and the goals are assumed to be known and the same for all agents. It is assumed that all agents want to collaborate to achieve the common goals.

Furthermore, agents may have different capabilities in which case the assignment of actions must be to agents that are capable of performing them. Capabilities may be overlapping, meaning actions can be performed by more than one agent, thereby making the assignment of actions to agents a non-trivial task.

Another important issue is authority. While there may be agents that have authority over other agents and thus can assign actions to those agents, we will not assume that there is a single agent that has authority over all others. This means the planning problem cannot be solved by having one agent plan for all others. Only those agents with authority over other agents will be allowed to plan for them, and in that case the problem can be reduced to one in which only the sub-group of agents that (between them) have authority over all others need to be considered, where this sub-group is a group of peers.

The application that motivates this problem is crisis response and management. The scenario here is that a number of agencies want to provide relief after a disaster has struck. These agencies usually have different (but overlapping) capabilities and no authority over each other. Truthfulness and a collaborative attitude are assumed in this scenario.

The Three Cranes Example

The example we will use to illustrate the collaborative multi-peer planning problem is a variant of the Dock Worker Robots domain [Ghallab *et al.*, 2004, section 1.7]. Since we will look at a single location only there are no robots. At this location are three cranes `k1`, `k2` and `k3` and three piles `p1`, `p2` and `p3` on which are three containers `c1`, `c2` and `c3` as shown in figure 1.

The cranes represent the agents in our scenario and we assume that none has authority over any other, i.e. they are peers. What distinguishes the cranes are their capabilities.

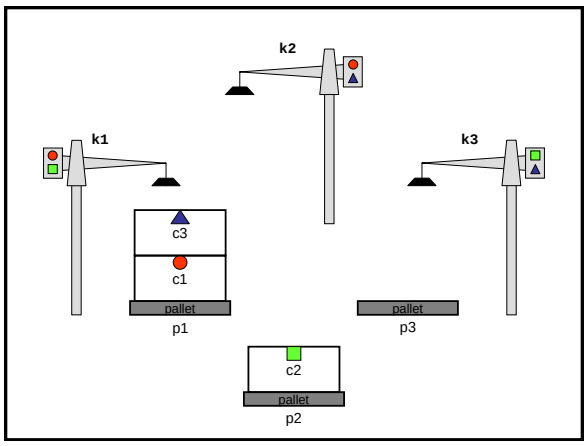

Figure 1: Initial state for the three cranes domain

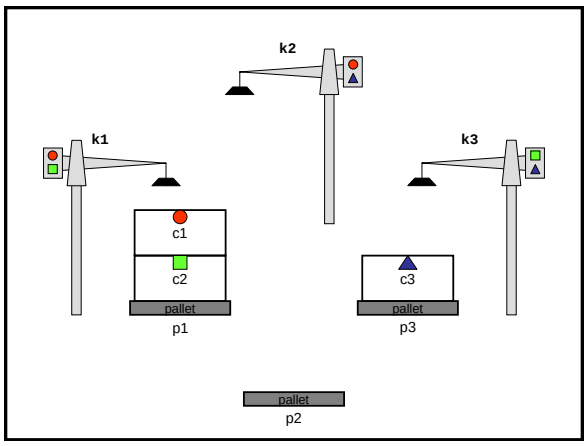

Figure 2: Goal state for a problem in the three cranes domain

Containers are of different types and not every crane can pick up every type of container. In this example, container `c1` is of type `t1` (red circle), container `c2` is of type `t2` (green square), and container `c3` is of type `t3` (blue triangle). Crane `k1` can lift containers of type `t1` and `t2`, crane `k2` can lift containers of type `t1` and `t3`, and crane `k3` can lift containers of type `t2` and `t3`.

Cranes advertise their capabilities to all other agents. For example, crane `k1` could advertise one of its capabilities (in a PDDL-like syntax) as:

```
(:activity move-container-t1
        :parameters (?c - container ?po ?pd - pile)
        :vars (?co ?cd - container)
        :precondition (and
                (type ?c t1)
                (in ?c ?po) (top ?c ?po) (on ?c ?co)
                (top ?cd ?pd) (/= ?po ?pd))
        :effect (and
                (in ?c ?pd) (top ?c ?pd) (on ?c ?cd)
                (not (in ?c ?po)) (not (top ?c ?po))
                (not (on ?c ?co))
                (top ?co ?po) (not (top ?cd ?pd))))
```

By advertising this capability crane `k1` informs the other agents that this is an activity it can perform. Internally the advertising agent may break down this activity into sub-activities to execute it, but the other agents can now treat the advertised capability as a primitive with which they can plan. However, since they have no authority over crane `k1`, it is not certain that the capability advertiser will perform the activity as it is not obliged to always perform advertised capabilities when they are requested. Note that `k1` is leaving out some detail in its capability description, namely that it must not be holding anything.

Crane `k1` will also advertise a second capability that is identical to the one above except for the first precondition which will be `(type ?c t2)`. Similarly, the other cranes will advertise capabilities for the respective container types.

Now, given the initial state depicted in figure 1 and the six activities advertised as capabilities by the different agents, what is missing from a definition of a planning problem is a goal state. One possible goal state is given in figure 2.

Agreed Plans

Note that it is not difficult to solve this problem as a classical problem. That is, it is simple enough to find a sequence of actions that transforms the initial state into the goal state. However, for the problem instance above every solution must involve more than one crane and since no crane has authority over any other crane, no crane can come up with a plan that it can ensure will be executed. Other agents may refuse the activities assigned to them, or multiple agents may want to execute the same activities that are required to get to a given goal state.

The heart of the distributed planning problem is therefore not the finding of a plan, but the finding of an *agreed* plan. Informally, by an agreed plan we mean a classical plan in which for every action in the plan there is an agent to which this action is assigned. Furthermore, this agent must be committed to the actions assigned to it in the plan, that is, it intends to execute these actions as required by the plan. Achieving this commitment is usually not considered part of the planning problem, but for the planning problem with multiple peers it is vital.

<I-N-C-A> and Task-Centric BDI

In this section we will formally define the concepts that form the basis for our definition of the collaborative multi-peer planning problem and, more importantly, what constitutes a solution to this problem. We will begin with activities that form the basic ingredients of a plan. Based on this we will define activity networks as objects in the <I-N-C-A> ontology [Tate, 2003]. Activity refinements provide a way of breaking an activity down into sub-tasks. This will provide a fairly standard HTN planning framework [Sacerdoti, 1975; Tate, 1977]. To extend this to the collaborative multi-peer

planning problem and finding agreed plans we need to define mental states of agents and we shall do so using a BDI approach [Rao and Georgeff, 1991]. Specifically, we will define different types of beliefs that agents have about the world and each other. Finally, we will briefly discuss the concept of capabilities as this is important for the planning as well as the mental states of agents.

Activities

Activities can be defined in a number of ways depending on what needs to be expressed in the planning domain. To ground our definition of the collaborative multi-peer planning problem we shall adopt a simple, STRIPS-like activity model. Other more expressive representations may be used here, e.g. ADL [Pednault, 1989]. What we mean by an activity schema corresponds to a STRIPS operator.

An *activity schema* consists of three components:

- s: the signature of the activity schema $n(v_1, \ldots, v_k)$ where
  - n is the unique name of the activity schema and
  - $v_1, \ldots, v_k$ are variables representing the parameters of the activity schema;
- C: the set of preconditions of the activity schema which can be literals in first-order logic or state variable assignments;
- E: the set of effects of the activity schema which can be literals in first-order logic or state variable assignments.

By an *activity* we mean a partially instantiated instance of an activity schema. That is, some of the variables representing parameters may be replaced by symbols referring to objects in the domain and there may be more than one instance of the same activity schema in a plan. We can use the signature plus a unique label to refer to activities.

By an *action* we mean an activity that is fully instantiated, i.e. all the parameters are ground.

If o is an activity schema then we use $o.s$ to refer to the signature of the schema, $o.C$ to refer to the preconditions of the schema, and $o.E$ to refer to the effects of the schema. This notation disambiguates the meaning of overloaded symbols. In general, we will use lower case letters for components that are objects of some kind whereas upper case letters are used for components that are sets of things.

For example, the activity of taking a container of type `t1` from the top of a pile could be described in as follows:

```
(:activity take-t1
        :parameters (?c - container ?po - pile)
        :vars (?co - container)
        :precondition (and
                (type ?c t1)
                (in ?c ?po) (top ?c ?po) (on ?c ?co)
                (empty ?agent))
        :effect (and
                (not (in ?c ?po)) (not (top ?c ?po))
                (not (on ?c ?co)) (top ?co ?po)
                (holding ?agent ?c)))
```

Notice that there is an implicit variable `?agent` defined for every activity in a multi-agent planning domain that is always bound to the agent performing the activity (or being responsible for its performance).

Activity Networks: <I-N-C-A>

Activities are organized in networks to form plans. We consider such a network to consist of 4 components described by an <I-N-C-A> object:

- I: the set of issues in the activity network, e.g. flaws or opportunities;
- N: the set of activities in the network (at all levels of refinement);
- C: the set of constraints associated with the network, e.g. ordering, time or resources;
- A: a set of annotations, e.g. rationale.

If p is an <I-N-C-A> activity network we use $p.I$, $p.N$, $p.C$, and $p.A$ to refer to the issues, activities, constraints, and annotations of this plan respectively. A more detailed and formal description of the <I-N-C-A> framework can be found in [Wickler *et al.*, 2006].

Activity Refinements

Refinements are used to break down high-level activities that are part of an <I-N-C-A> activity network into (primitive) actions in HTN planning. An activity refinement consists of the following components:

- n: the unique name of the refinement;
- a: the signature of the activity to be refined (also called the task);
- A: the set of sub-activities that constitute the refinement;
- C: some constraints on activities in A.

If there is no refinement available for a given activity then we assume that this activity is *primitive* and there should be an agent that can execute this activity. If there is a refinement available for a given activity we consider this activity *abstract* and it needs to be refined before it becomes executable.

For example, we can break down the abstract activity `move-container-t1` described above into (primitive) sub-activities using the following refinement:

```
(:refinement take-and-put-t1
        :activity (move-container-t1 ?c ?po ?pd)
        :vars (?crane - agent)
        :sub-activities (
                (1 ?crane (take-t1 ?c ?po))
                (2 ?crane (put-t1 ?c ?pd)))
        :constraints (
                (before 1 2)))
```

Beliefs Desires and Intentions

Activities, activity networks, and activity refinements together form an HTN-style planning framework that does not treat agents in any special way. For the collaborative multi-peer planning problem we need to define the internal structure of an agent in more detail (see also [Wickler *et al.*, 2007]) and we shall assume a BDI-like framework here. That is, the mental state of an agent can be described by a set of beliefs (B), desires (D), and intentions (I).

Both, desires and intentions, are represented as <I-N-C-A>-objects, i.e. plans, in our approach. This is simply to fit in with our HTN planning framework. Desires include all those activities that an agent would like to perform at some point in the future. These activities can be abstract or primitive, and they may or may not executable.

Intentions include activities that an agent is committed to performing. This commitment may be to another agent or it may just be internal. Non-primitive activities must be refined by the planner until all intentions are primitive activities and the agent believes them to be executable (when they are due to be executed). Intensions need not be immediate, that is, they are usually scheduled for an execution time in the future.

Belief Types

To define the collaborative multi-peer planning problem, the beliefs of an agent must include various types of beliefs about the activities that can be performed by the different agents defined in the planning problem. Thus, we shall divide the beliefs into the following components:

- S: the usual knowledge about the world state, which will mostly be used to verify the preconditions for activities and refinements are satisfied;
- C: the set of capability descriptions known to the agent, i.e. a set of pairs (a, o) where a is the agent that has advertised the capability and o is the signature of an activity schema that describes the advertised capability;
- R: the set of activity refinements known to the agent;
- P: the set of primitive activities that this agent can execute (and knows that it can).

For example, in the initial state given above, S would contain all the logical atoms that are true in this state. The capability descriptions would be the same for all agents and would consist of six capabilities, two for each crane, e.g. for crane `k1` which can lift containers of type `t1` and `t2`:

```
(:capability
      :agent k1
      :activity (move-container-t1 ?c ?po ?pd))
(:capability
      :agent k1
      :activity (move-container-t2 ?c ?po ?pd))
```

Depending on the expressiveness of the formalism used these may be represented as a single capability containing a disjunction.

The refinements R would be a set of refinements as described above. The set of primitive activities are different for the agents in our example as each agent only needs to know the activities it itself can execute. For example, crane `k1` would have the following beliefs in P:

```
(:can-execute
      :activity (take-t1 ?c ?po))
(:can-execute
      :activity (put-t1 ?c ?pd))
(:can-execute
      :activity (take-t2 ?c ?po))
(:can-execute
      :activity (put-t2 ?c ?pd))
```

Capabilities

The notion of capabilities described so far is actually too simplistic for practical applications, but it is sufficient for the three-cranes toy problem used here. It is assumed that capabilities are described using a shared ontology of activity descriptions. Since an agent associates only the signature of an activity with another agent in a capability description, there needs to be a description associated with that signature that all agents agree on. A hierarchical ontology of activities would be more useful for this purpose than the simple model described above.

Also, it is not realistic to assume that an agent will advertise as part of its capability description all the constraints that it considers preconditions for the application of this capability. For example, one precondition that applies to every capability description is that the agent is not already otherwise committed. That is, agents that can only perform one action at a time may in principle be capable of performing an advertised capability, but they may not be able to perform it at any requested time. As commitments are expected to change relatively frequently but advertised capabilities are expected to remain constant, such scheduling constraints should not be part of a capability advertisement.

Another practical concern is that agents usually only perform their advertised capabilities as part of a larger protocol which in itself can be seen as a plan. For example, an agent might expect to be paid for performing its capability and there may be actions that correspond to that process that the capability requester needs to perform. For the example used here we shall assume that all capabilities are independent of other actions.

Formalizing the Collaborative Multi-Peer Planning Problem

We can now formalize the collaborative multi-peer planning problem. In addition to the traditional components of a planning problem (operators, initial state, and goal) this requires a set of agents with their mental (BDI) states as described above. The next step must then be a definition of what constitutes a solution to such a problem. This will define the semantics of the problem.

The Problem Specification

A collaborative multi-peer planning problem is given by a pair (A, N) where A describes a set of agents and N is an initial activity network.

The first component, $A = \{a_1, \ldots, a_n\}$, consists of a set of agent descriptions where each agent a_i is defined by its beliefs $a_i.B$, its desires $a_i.D$, and its intentions $a_i.I$ as described above. Note that the beliefs include beliefs about the world state, the capabilities of the agent itself as well as other agents, available refinements for breaking down tasks, and primitive activities the agent can execute. Having refinements as part of the agents' knowledge means they do not need to be listed explicitly as part of the planning problem.

The second component N is an <I-N-C-A> activity network consisting of issues $N.I$, activities (nodes) $N.N$, constraints $N.C$, and annotations $N.A$. In this HTN-style specification of a planning problem it is not necessary to list the initial state and the goal separately as they can be defined in the initial network with two dummy activities `init` and `goal` as is usual in HTN planning.

We shall assume that there are no inconsistencies in the planning problem and that all agents have complete knowledge of each other's capabilities. The former implies that the initial state implicit in N and the agents' beliefs about the current world state $a_i.B.S$ are equivalent. The latter implies that for all agents a_i, a_j their beliefs about capabilities are the same, i.e. $a_i.B.C = a_j.B.C$.

Solutions

The above definition of a collaborative multi-peer planning problem is not significantly different from classical HTN planning, except that the agents that execute actions are made explicit. Note that this already creates a different solution space if one assumes that agents can only execute one action at a time.

The solutions we are looking for must include an important additional condition: they must be agreed by all the agents in the problem. This can be formalized by requiring that every action that is part of the plan and is assigned to an agent must be an element of that agent's intentions, i.e. every agent is committed to executing its share of the plan as a result of the planning process.

An agreed solution to a collaborative multi-peer planning problem (A, N) is a pair (N', s) where:

- N' is a ground, primitive activity network and
- s is an assignment of activities in N' to agents in A

such that:

- there exists a decomposition tree $Delta$ from N to N'
- every action is assigned to an agent, i.e. $\forall a \in leafs(\Delta) : s(a) \in A$
- assignments are to agents capable of performing the activities, i.e. $\forall a \in leafs(\Delta) : a \in s(a).B.P$
- agents intend to perform activities assigned to them, i.e. $\forall a \in leafs(\Delta) : a \in s(a).I$

An additional condition that could be included here but which is not strictly necessary is to include the assignment of abstract activities in the decomposition tree to agents, and require assignments to be only to agents that advertise a matching capability.

An Approach: Auctions

We will now discuss a solution approach to the collaborative multi-peer planning problem which draws some parallels to the idea behind auctions as a method for solving the problem of finding an agreed price.

Auctions and Agreement

One way of viewing an auction is as a process for agreeing on something. A group of agents want to buy an item one of them has to sell, and they need to agree who will buy the item and at what price. To do so, they hold an auction. That is, they first agree on a process they will execute and that will result in the agreed price and buyer. Then they execute that process and have the desired agreement. Thus, instead of directly seeking the agreement they are after, they first agree on a process that leads to the agreement, and then following this process. If every agent agrees on the process, finding the desired result is no longer difficult.

Finding Agreed Plans

The question then is what such a process could look like. One possible procedure is based on the idea of each agent solving the overall problem as if it had authority over the other agents, then negotiating additional constraints that may not have been part of the capability descriptions, and finally voting for the agreed plan:

```
repeat:
        every agent:
                solve HTN planning problem
                suggest plan that self would agree to
        for every plan:
                every agent:
                add constraints on own activities
until no agents add further constraints
vote for solution plan; adopt it
```

The agents start by solving the planning problem as a classical planning problem, treating all capabilities as available primitive activities. This may seem like a waste of computational resources but it is unlikely that agents will accept other agents' plans without thinking about the way in which the common goal can be achieved themselves. Furthermore, HTN problems tend to be solvable without too much effort for most practical domains for which there is a known set of standard operating procedures.

In the next step agents can suggest the solutions they would like to see implemented to the other agents, e.g. using a blackboard architecture. For example, eager cranes might want to suggest plans in which they do as much work as possible. There is no requirement for every agent to suggest a plan; they may just rely on others, e.g. if there is a peer they fully trust. In the worst case there could be no suggested plans and this simply reflects the fact that there might not be any solution plans to the classical problem, just like there might not be an agreeable price for an auction.

Once the plans have been suggested, agents can criticize each others plans. This can be done by adding further constraints to the activities in the plan. For example, an agent

may be assigned an activity that conforms with its advertised capabilities as part of some other agent's plan, but it may not be able to perform the capability at that time or in the specific circumstances. Adding such a constraint at this stage will invalidate the plan and the agent who suggested the plan may then re-plan in the next iteration of the loop.

This is continued until no agent adds further constraints to any of the plans, in which case the set of candidate plans is hopefully not empty. Note that each of these plans represents a valid classical solution. Now the agents need to choose one of these solutions and this can be achieved by voting. If this procedure was agreed the selected plan should now be adopted by all agents and thus, the collaborative multi-peer planning problem is solved.

Conclusions

The main contribution of this paper lies in the definition of a new kind of distributed, multi-agent planning problem, the collaborative multi-peer planning problem. In this problem a group of peers (agents without authority over each other) have to agree on a plan that achieves a common goal. Agents are defined with BDI-style mental states, and a plan is considered agreed if the actions in the plan are assigned to agents that are committed to executing these actions. This commitment is an important aspect that needs to be part of the outcome of the planning process. The proposed approach introduces a level of indirection into the planning process—instead of agreeing the plan directly the peers agree on the procedure for finding the plan and then execute this procedure.

The proposed approach uses planning techniques to generate candidate solutions and a combination of negotiation and voting for achieving agreement. A simple combination of planning and voting is not considered to be a solution (at least in our emergency response scenario) as the negotiation aspect effectively gives agents the option of vetoing any plan that involves them. Note that, for the multi-peer planning problem, there may exist a plan that solves the planning problem, but there may not be a plan that all the agents can agree on. However, if the negotiation phase results in a non-empty set of plans, this means that all agents involved have no further objections against any of those plans, and the voting is only intended as a means for choosing one of the agreeable plans.

There are some remaining issues concerning the negotiation phase, however. The current solution allows only for a limited set of criticisms that can be expressed, but agents may want to object to proposed solutions on more general grounds, e.g. an "unfair" distribution of work. Part of the problem here seems to be that the subject of negotiation changes from the a plan to an optimization criterion or a preference. Similarly, if there is no agreed plan, the negotiation could be about which goals to change or drop. The solution presented here does not account for such negotiations.

Future work will look at how decommitment can be incorporated into the solution. The idea here that it will be easier to agree on a plan if agents can specify conditions under which they are allowed to drop a commitment.

References

James Allen, James Hendler, and Austin Tate, editors. *Readings in Planning*. Morgan Kaufman, 1990.

Edmund H. Durfee. Distributed problem solving and planning. In Gerhard Weiss, editor, *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*, chapter 3, pages 121–164. The MIT Press, 1999.

Malik Ghallab, Dana Nau, and Paolo Traverso. *Automated Planning*. Morgan Kaufmann, 2004.

Edwin Pednault. ADL: Exploring the middle ground between STRIPS and the situation calculus. In *Proc. 1st International Conference on Knowledge Representation and Reasoning (KR)*, pages 324–332. Morgan Kaufmann, 1989.

Anand Rao and Michael Georgeff. Modeling rational agents within a BDI-architecture. In *Proc. 2nd International Conference on Knowledge Representation and Reasoning (KR)*, pages 473–484. Morgan Kaufmann, 1991.

Earl D. Sacerdoti. The nonlinear nature of plans. In *Proc. 4th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 206–214. Morgan Kaufmann, 1975. Reprinted in [Allen *et al.*, 1990, pages 162–170].

Austin Tate. Generating project networks. In *Proc. 5th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 888–893. Morgan Kaufmann, 1977. Reprinted in [Allen *et al.*, 1990, pages 291–296].

Austin Tate. <I-N-C-A>: A shared model for mixed-initiative synthesis tasks. In Gheorghe Tecuci, editor, *Proc. IJCAI Workshop on Mixed-Initiative Intelligent Systems*, pages 125–130, 2003.

Gerhard Wickler, Stephen Potter, and Austin Tate. Recording rationale in <I-N-C-A> for plan analysis. In Lee McCluskey, Karen Myers, and Biplav Srivastava, editors, *Proc. ICAPS Workshop on Plan Analysis and Management*, pages 5–11, 2006.

Gerhard Wickler, Stephen Potter, Austin Tate, Michal Pěchouček, and Eduard Semsch. Planning and choosing: Augmenting HTN-based agents with mental attitudes. In *Proc. International Conference on Intelligent Agent Technology*, 2007.