Distributed Belief Revision

Benedita Malheiro, Eugénio Oliveira and Nick Jennings

Benedita Malheiro

Dept. of Electrical Engineering, Instituto Superior de Engenharia do Porto, Rua de S. Tomé, 4200 Porto, Portugal.

Telephone: +351-2-82 11 34

Fax: +351-2-82 11 29

Email: mbnm@fe.up.pt

Eugénio Oliveira

Dept. of Electrical Engineering and Computers, Faculty of Engineering, Opporto University, Rua dos Bragas - 4099 Porto CODEX, Portugal.

Telephone: +351-2-200-7505

Fax: +351-2-319-280

Email: eco@fe.up.pt

Nick Jennings

Department of Electronic Engineering, Queen Mary & Westfield College, University of London, Mile End Road, London E1 4NS, UK.

Telephone: +44-71-975-5349

Fax: +44-81-981-0259

Email: N.R.Jennings@qmw.ac.uk

Content Areas: Distributed Reasoning and Control; Cooperation; Belief Revision.

Abstract

The ability to respond sensibly to changing and conflicting beliefs is an integral part of intelligent agency. To this end, we outline the design and implementation of a Distributed Assumption-based Truth Maintenance System (DATMS) appropriate for controlling cooperative problem solving in a dynamic *real world* multi-agent community. Our DATMS works on the principle of *local coherence* which means that different agents can have different perspectives on the same fact provided that these stances are appropriately justified. The belief revision algorithm is presented, the meta-level control actions needed to ensure that all system-wide queries can be uniquely answered is described, and the DATMS' implementation in a general purpose multi-agent shell is discussed.

Distributed Belief Revision*

Benedita Malheiro, Nick Jennings and Eugénio Oliveira

1. INTRODUCTION

In many AI applications problem solving entities often have to make decisions based on partial, imprecise, and ever changing information. However in systems in which several agents cooperate with one another within a decentralised control regime, the information management problem is exacerbated still further - each agent has to contend with deficiencies and changes in the information supplied by its contemporaries as well as in its own local information.

To keep track of an agent's changing beliefs, researchers have devised a number of different types of Truth Maintenance System (TMS) [Martins, 1990]. Such systems portray as their main features the maintenance of the coherence between their beliefs, the reason for their beliefs and the identification of contradictions. Whilst these systems are generally sufficient for maintaining beliefs in an asocial context, they need to be extended if they are to be used in a social context. For example, as well as beliefs that an individual has generated for itself, there will be beliefs that it has been informed about by other community members (either because an acquaintance has answered a query or because it has volunteered a piece of relevant information). In such cases a number of crucial decisions must be made about how the information provided by other agents should be treated - should it be given the same credence as locally deduced beliefs?, should it only be used when there is supporting local evidence?, how should contradictions between the beliefs of different agents be dealt with?, and so on. It is in the discussion and resolution of these issues (section 2), and their subsequent implementation (section 3), that the main contribution of this work lies.

2. REVISING BELIEFS IN MULTI-AGENT SYSTEMS

This section briefly describes the fundamentals of belief revision systems (section 2.1), before the key principles and concepts of belief revision in multi-agent systems are expounded (section 2.2).

2.1 Asocial Belief Revision

In a TMS "belief" is taken to mean justified belief - either it is an assumption or it has been deduced from other beliefs. While an assumed facts is believed, an ordinary fact may be believed or unbelieved; and a contradiction is always false. There are a number of different ways in which the dependencies between beliefs can be registered: in justification based TMSs (JTMSs) each belief is associated with the beliefs that immediately caused it [Doyle, 1979]; whereas in an assumption based TMS (ATMS) each belief is associated with the smallest set of environments from which it can be deduced (*the belief's label)* [de Kleer, 1986a]. This work concentrates on the ATMS approach because, when compared with the JTMS, it:

- enhances the system's efficiency and improves its real time operation because multiple contexts are kept. This makes it faster to move from one valid context to another because the new one does not have to be calculated from scratch.
- improves the system's transparency because the belief revision is not contingent on dependency oriented backtracking and hence it will not stop on unresolved circularities or leave nodes unlabeled.

ATMS are composed of three units: (i) the truth maintenance system itself; (ii) the problem solver; and (iii) the interface unit. The TMS guarantees that the conclusions reached by the

* This work has been supported by the BritishCouncil/JNICT Protocol Grant 423

problem solver are kept updated and coherent. However it only deals with propositions (usually substituted by arbitrary identifiers called nodes) and their dependencies. For each proposition there will be a node and for each dependency a justification which describes how the node was deduced from other nodes. The TMS has three basic operations: (i) create a new *assumption node* whenever new propositions are assumed to be true; (ii) create a new *ordinary node* when a new proposition is deduced by the problem solver; and (iii) add a new justification to an existing node whenever the problem solver finds a new way of deducing it.

While a conventional rule-based problem solver generally develops a unique solution, a reasoning system with an ATMS determines the whole set of possible solutions. This different operating mode necessitates the adoption of a different problem solving methodology and a different knowledge representation. The rules have to be transformed into sets of unit steps called consumers. This transformation includes the calculation of the consumer's depth which indicates its dependence. The greater this depth, the greater the consumer's dependence on others (i.e. execution of a consumer of depth D can only be done after the D-1 depth consumers have been executed, and so forth). Using this depth information, the problem solver chooses the nodes according to the number of assumptions they depend on, starting with the nodes with the fewest assumptions. A consumer is triggered only when the ATMS holds valid justifications for each of its precondition nodes; once fired, it converts itself into a justification of the conclusion node and ceases to exist as a consumer. The set of consumers to be executed at any given time can be obtained through a scheduling algorithm [de Kleer, 1986b] and an agenda which contains every node with a non empty label and pending consumers. The problem solver repeatedly chooses one of these consumers, executes it, and then removes it, until there are no nodes left on the agenda.

The main purpose of de Kleer's algorithm is to find, as efficiently as possible, the most general environment of a node and the most general version of a contradiction. Adopting this strategy avoids unnecessary work since: (i) by finding the most general version of a contradiction it avoids various solving steps which would only lead to inconsistencies; (ii) by finding the most simple node label it avoids superfluous label updating. This approach guarantees that the consumers of the antecedent nodes will always be scheduled before the node's own consumers. The main cycle consists of choosing the consumer with the smallest consistent environment, running it, and then removing it. With such an agenda, the ATMS-problem solver combination will always find the most simple labels first - identifying all possible solutions with the least effort.

The final component is the interface unit which supports the following functionalities: (i) assumption creation; (ii) node creation; (iii) addition of justifications; and (iv) querying about beliefs.

2.2 Locally Coherent Multi-Agent Belief Revision

Ideally a belief revision system should deliver complete coherence between all of the agents' conclusions all of the time. However attainment of this level of coherence depends not only on the system's architecture and design but also on the amount of inter-agent communication which is acceptable. Whenever a centralized architecture is appropriate it is reasonable to build a global TMS which incorporates all of the system's facts and justifications; whereas in the case of a distributed architecture a pragmatic compromise between the achieved coherence level and the information redundancy among the agents has to be reached. These two architectural options give rise to two fundamental approaches to belief revision in a multi-agent system: respectively, *global coherence* and *local coherence*. In the first case, two or more agents cannot assign a different belief status to the same fact. In the second case, different agents may have different perspectives over the same fact if conveniently justified. For example, provided they both have the appropriate justifications, agent$_1$ may believe that the fault is in region$_1$ of the network, whereas agent$_2$ may believe that the fault is in region$_2$.

In multi-agent systems the autonomous agents each have their own repositories where they record local propositions and justifications. Only when cooperation occurs do non-local facts have to be represented. In such an environment global coherence is unattainable, unless the system broadcasts every relevant activity to all the pertinent agents, therefore we settled for local coherence.

Given this stance, a new crucial issue arises: the question of how to include external propositions in an agent's local dependency network. Depending on the scheme chosen for attaining local coherence, an agent that receives an external fact may or may not receive its label: if the label is sent, it is possible to guarantee the coherence between the foundations of the external fact and the local facts and assumptions; if no label is sent, it is impossible to cross check the external fact's foundations with the local TMS data. In the first case the agents exhibit *local-and-shared well-foundedness* and local coherency, whilst in the second case there is only *local well-foundedness* and local coherency [Huhns, 1991]. We chose the latter because we believe it is more appropriate for modelling real-time autonomous agents that have their own beliefs, desires and intentions. Consequently, the community of agents behaves like a democratic society in which each individual can hold a different opinion once it is locally justified - an agent only accepts to revise its beliefs based on external information when it does not have its own convictions regarding that fact.

Using this scheme, a given agent's knowledge has to be divided into two separate sets:

- *private beliefs* that the agent has generated and kept to itself.
- *shared beliefs* that the agent has in common with at least one acquaintance.

Within a particular agent, a shared belief can either be *internal* or *external*. The former means that the agent has deduced the fact for itself; the latter means that the agent has received the information from an acquaintance. Thus in the fault diagnosis scenario presented earlier, for example, the fact that the fault is believed to be in region1 is shared between agent$_1$ and agent$_2$, it is shared internal for agent$_1$ and shared external for agent$_2$. This classification is central to the belief revision process because:

- private facts have a local scope and are automatically revised by the agent's ATMS.
- shared facts are revised only by the agent that created them (i.e. agents where the facts are classified as shared internal). The revision action is performed by the originating agents' ATMS module and then the updated beliefs are resent to all the acquaintances where they are known as shared external.

The basic operation of each agent can be described as a continuous loop of: (i) receiving and processing new beliefs; (ii) updating the items on the agenda; (iii) executing its agenda. Whenever a fact is presented to the system it is immediately routed to the appropriate agents. The recipients are those agents where the fact has been, or can be, assumed or deduced. Once the arrived fact has been locally revised it is sent to all of the acquaintances where the necessary updates take place. In more detail, the belief revision algorithm followed by each agent is given below:

```
IF a fact F has arrived at AGENT
THEN
    IF F EXISTS IN AGENT
    THEN
        IF SHARED-EXTERNAL(F)
        THEN
```

```
                IF incoming belief status equals existing one
                THEN nothing is done
                IF F's status is unbelieved or false & it is
                currently believed by AGENT
                THEN invoke ATMS to remove assumption that
                represented F and create an ordinary node with
                an empty label
                IF F's status is believed & it is currently
                unbelieved by AGENT
                THEN invoke ATMS to remove the ordinary node
                with an empty label that represented F and
                create an assumption
           IF SHARED-INTERNAL(F)
           THEN
                IF F sent by an agent other than user agent
                THEN nothing is done
                IF F's new status ≠ AGENTs current belief
                status
                THEN invoke ATMS in standard manner
      IF F DOES NOT EXIST IN AGENT
      THEN
           IF F's belief status is unbelieved or false
           THEN create an ordinary node with an empty label
           IF F's belief status is believed
           THEN create an assumption
After this processing, the AGENT updates its own agenda and
executes it. For every node in the agenda the ATMS is invoked in
the standard manner, that is:
      IF node has pending consumers & node's antecedents are
      believed
      THEN incoming node's justification is sent to ATMS
Finally, if any shared internal nodes are revised during these
actions then their incoming belief statuses are sent to the
relevant acquaintances.
```

This algorithm does not guarantee that a unique belief status will be attributed to every shared fact. If the domains of several agents overlap, then a shared fact can be internal to several community members. In such circumstances, whenever a belief revision occurs in any of these agents, the system may find that the same fact has a different belief status in different agents. The way in which the agents that rely on such shared external facts decide which belief status to adopt is explained in section 3.3.

bn3. IMPLEMENTING LOCALLY COHERENT MULTI-AGENT BELIEF REVISION

The aforementioned multi-agent belief revision algorithm has been implemented in a general purpose system which facilitates cooperation between autonomous problem solving agents. The agents themselves are divided into two distinct functional units: a domain level system and a cooperation layer. The former is implemented as an assumption based belief revision system and contains, among others, expertise on diagnosing faults, planning and scheduling (section 3.1). The latter is the interface between the agent and the rest of the community and provides the necessary facilities for establishing, maintaining and monitoring cooperation (section 3.2).

3.1 Implementing Distributed Belief Revision

As a first step, the rules and facts of the domain level system's knowledge base (KB) are transformed into consumers and nodes. The knowledge representation used for the nodes and the consumers is as follows:

- `consumer(Dep, Id, List_of_Antecedents, Consequent)` in which Dep stands for depth, Id for the identification of the rule that originated that specific consumer, List_of_Antecedents contains the precondition nodes list and Consequent contains the conclusion node.
- `node(Datum, Classification, Type or Index, Scope, Agent)` where Datum is the proposition the node represents, Classification specifies whether the node is an assumption or a deduced node, the third argument, if the node represents an assumption, contains its index, otherwise it specifies the deduced node type, Scope specifies whether the proposition is local, shared internal or shared external, and Agent contains the name of the agent responsible for having deduced the node.

3.2 Implementing the Cooperation Layer

The functionality related to cooperation is represented as a distinct problem solving layer which sits above the ATMS+problem solver layer (figure 1). The cooperation layer has the following components [Wittig, 1992]: (i) a cooperation module; (ii) a communications module which sends messages between the agents; (iii) a self model which represents information about the underlying domain level system; and (iv) a set of acquaintance models which represent the relevant information about the other community members with which the agent can be expected to interact.

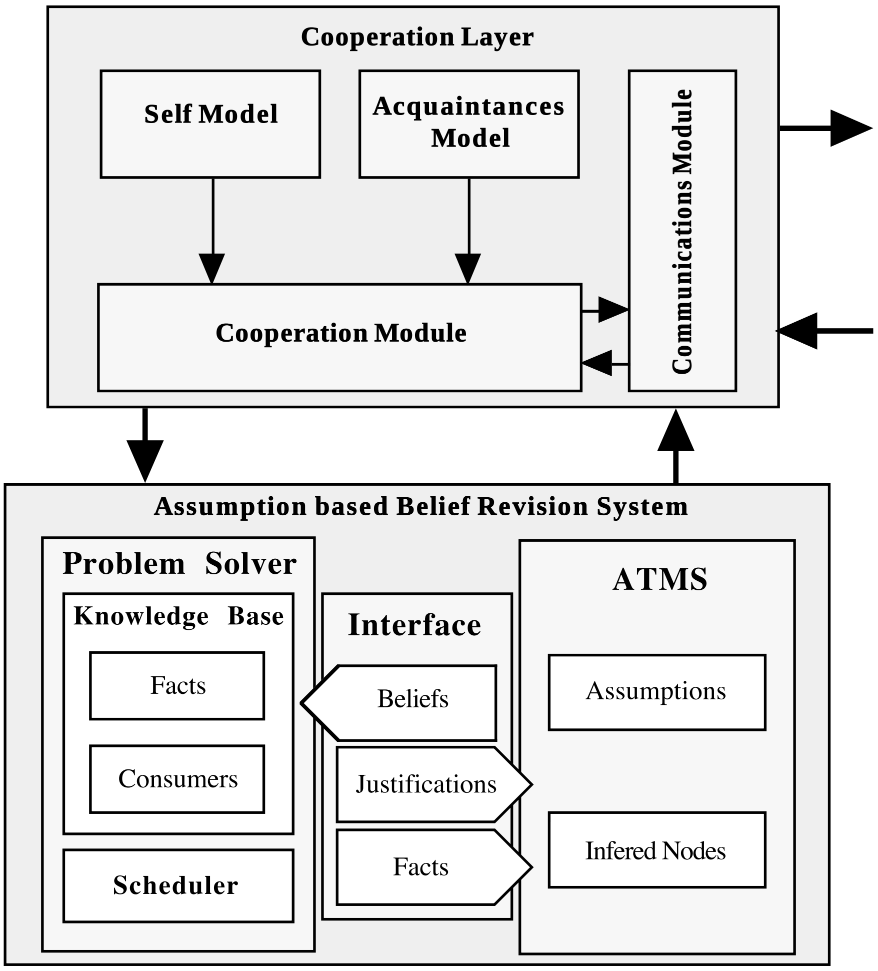

Figure 1: Agent Architecture

The Cooperation Module is responsible for determining when cooperation is necessary and for deciding what strategy should be employed for each social interaction. Cooperation can be viewed from two perspectives: from the organiser's point of view and from the respondent's point of view [Oliveira, 1993]. The organiser starts the cooperation based on its needs and views. This includes asking for assistance (*task sharing*) and supplying voluntary help (*result sharing*). Task sharing is initiated when the organiser has an activity that it cannot accomplish alone and so it looks for help within the community - the agent which accepts the task is the respondent. Result sharing is initiated when the organiser generates information which it believes will be useful to others, based on its acquaintance model, or if it has revised some of its shared beliefs.

The Communications Module provides the necessary physical channels and the high-level protocol needed to support cooperative problem solving. The implemented system is based on message passing through the UNIX Operating System sockets. The high-level protocol is based on speech act theory and has the following primitives:

- `request(Organiser, Respondent, Data)`: Organiser asks the Respondent for help in producing Data (task sharing).
- `query(Organiser, Respondent, Data)`: Organiser queries the Respondent about a belief.
- `answer(Respondent, Organiser, Data)`: Respondent answers a request within the framework of a task sharing cooperation.
- `assign(Organiser, Respondent, Data)`: Organiser initiates a result sharing cooperation action as the result of some shared beliefs' revision.
- `reply(Respondent, Organiser, Data)`: Respondent answers a previous query about a belief.
- `explain(Respondent, Organiser, Data)`: Respondent explains the reason for a particular belief, e.g., provides its foundations.

In order to participate effectively in a multi-agent system, an agent must know which of its tasks can be carried out without assistance (*independent tasks*) and which of its tasks depend on other agents in some way (*dependent tasks*). This information is represented in the agent's self model: `who_am_i` gives the agent's identification; `my_conclusions` gives the tasks the agent is capable of performing; `i_know_about` gives the facts the agent knows about; and `my_goal` gives the requirements and results of each task.

The acquaintance models provide agents with information about who can assist them with their dependent tasks (`who_knows_about`) and who they can assist by volunteering useful information (`is_interesting_to`).

3.3 Cooperative Problem Solving

A cooperative interaction is started: (i) when an agent needs assistance; (ii) when an agent is able to supply help; and (iii) when a belief revision of shared knowledge occurs. An example of each of these cases is presented below:

i) Task Sharing Cooperation:

Agent$_1$ requests help from agent$_2$ to determine whether michael is a member of staff. Agent$_1$ knows that it is unable to produce this information by examining the `my_conclusions` slot of its self model, the `who_knows_about` slot of its acquaintance model indicates that agent$_2$ can assist it in this activity.

`request(agent`$_1$`, agent`$_2$`, staff-michael)`

If agent$_2$ can deduce that michael is a member of staff it will answer agent$_1$ by sending the corresponding belief status (eg believed).

`answer(agent`$_2$`, agent`$_1$`, staff-michael-believed)`

Then the already detailed belief revision algorithm is locally triggered at $agent_1$ with staff-michael-believed as a shared external fact.

ii) Result Sharing Cooperation:

$Agent_1$ sends voluntarily a result (e.g. that john is believed to have been promoted) to $agent_2$. This exchange occurs because $agent_1$'s model of $agent_2$ states in the `is_interesting_to` slot that john's promotion is a relevant piece of information.

`assign(agent`$_1$`, agent`$_2$`, promoted-john-believed)`

Then the already detailed belief revision algorithm is locally triggered at $agent_2$ in order to incorporate this new proposition as a shared external fact.

iii) Belief Revision of a Shared Fact:

(i) $Agent_1$ no longer believes in the internal shared fact that john has been promoted and through consulting its acquaintance models (the `is_interesting_to` slot) it passes this information onto $agent_2$.

`assign(agent`$_1$`, agent`$_2$`, promoted-john-unbelieved)`

(ii) $Agent_2$ no longer believes in the internal shared fact that michael is a member of staff and through consulting its acquaintance models (the `is_interesting_to` slot) it passes this information onto $agent_1$.

`assign(agent`$_2$`, agent`$_1$`, staff-michael-unbelieved)`

Then, in both cases, the already detailed belief revision algorithm is locally triggered at the receiver agent in order to update the corresponding shared external belief.

The adopted architecture results in locally responsible agents that provide the community with local coherence. However, this approach still leaves some important questions unanswered: what does the system answer when it is queried about a fact for which different agents have a different belief status? and what belief status will be adopted by an agent that has represented such a shared external fact? With respect to the first question, there are two possible situations:

- there is at least one agent in the community to which the fact is a contradiction.
- there is no agent in the community to which the fact is a contradiction.

In the first case, the system answers that the fact is a contradiction and therefore it is false; in the second case, if there is at least one agent that has reasons to believe in the fact, the system answers the query by stating that the fact is believed. Essentially the same mechanism has also been adopted to solve the second question of deciding in which acquaintance an agent should believe. Some meta-level control actions are included on top of the already described belief revision algorithm to act as a filter for the incoming data:

```
IF incoming fact F is a new fact
THEN call the standard belief revision algorithm
IF F already exists
THEN
     IF the agent's belief status for F is false
     THEN nothing is done
     IF the agent's belief status for F equals the incoming
     belief status for F
     THEN nothing is done
     IF the agent's belief status for F is believed & the
     incoming belief status is unbelieved
     THEN nothing is done
     IF the incoming belief status is false (F is a
     contradiction to some agent)
```

```
THEN F will be revised according to the incoming data
(becomes false and will be owned by the agent that found
the contradiction)
IF the agent's belief status for F is unbelieved & the
incoming belief status is believed
THEN F will be revised according to the incoming data
(unbelieved -> believed)
```

4. CONCLUSIONS

This paper outlines the key issues associated with belief revision in multi-agent systems and describes how a locally coherent cooperating community for real world scenaria can be designed and implemented. The adopted implementation platform, which was originally conceived as a general purpose multi-agent architecture supporting different forms of cooperation [Wittig, 1992], also turned out to be ideally suited for supporting belief revision activities. The knowledge contained in the cooperation layer's models ensures that if a change occurs in an existing domain level system fact it will be communicated to every agent where it is external, additionally the models ensure that incoming facts are always communicated to the relevant agents.

Our distributed belief revision system guarantees a unique justified answer to every query with a time response adequate to real time applications. This is achieved through the selected architecture and the meta-level control actions.

Mason and Johnson implemented a Distributed ATMS where the interchange between the agents includes not only the shared data, but also the shared data labels and the invalid assumptions sets [Mason, 1989]. Their agents exhibit local and shared well foundedness (the foundations of the incoming facts are cross-checked with the local assumptions in order to guarantee that these foundations are not locally disbelieved) as well as local coherence. In our system, the agents exhibit only local well foundedness; when we decided not to send the label together with the node's belief status and the contradictions environments, we dropped the possibility of guaranteeing the shared facts well foundedness in order to gain in time. Instead, we apply the following methodology throughout our system: (i) every answer provided by the system is inspected in order to verify that it does not contain any of the system's detected contradictions;(ii) before adopting or updating a shared external belief the receiving agent collects all of the different perspectives regarding it, and only then asserts its status, based on the results of the described meta-level control actions.

Huhns and Bridgeland implemented a distributed JTMS that provides local and shared coherence [Huhns, 1991]. Their system suffers from the typical JTMS problems: computational overhead (not suitable for real time applications) and possible unsatisfiable circularities. In addition their implementation allows an agent with less information to dominate a more knowledgeable agent and their system is "agnostic about what data should be shared among agents".

We are currently investigating how our Distributed ATMS implementation can be applied in the domain of geographical information systems [Malheiro, 1994]. On the theoretical side a number of issues require further investigation, these include: explicit negation, negotiation and more sophisticated conflict resolution.

REFERENCES

[de Kleer, 1986a] J. de Kleer, (1986) "An Assumption-based TMS" Artificial Intelligence 28 (2), 127-162.

[de Kleer, 1986b] J. de Kleer, (1986) "Problem Solving with the TMS" Artificial Intelligence 28 (2), 197-224.

[Doyle, 1979] J. Doyle, (1979) "A Truth Maintenance System" Artificial Intelligence 12, 231-272.

[Huhns, 1991] M. N. Huhns and D. M. Bridgeland, (1991) "Multi-Agent Truth Maintenance" IEEE Trans. on Systems, Man and Cybernetics 21 (6), 1437-1445.

[Malheiro, 1994] B. Malheiro, (1994) "A Multi-Agent Geographical Information System" Technical Report, Dept. of Electronic Engineering, University of Opporto.

[Martins, 1990] J. P. Martins, (1990) "The Truth, The Whole Truth, and Nothing But The Truth" AI Magazine, Special Issue, 7-25.

[Mason, 1989] C. Mason and R. Johnson, (1989) "DATMS: A Framework for Distributed Assumption Based Reasoning" in Distributed Artificial Intelligence Vol II (eds. L Gasser and M. N. Huhns) 293-317.

[Oliveira, 1993] E. Oliveira and F. Mouta, (1993) "A Distributed AI Architecture Enabling Multi-Agent Cooperation" Proc. of 6th Int. Conf. on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, Edinburgh, UK.

[Wittig, 1992] T. Wittig (ed.), (1992) "ARCHON: An Architecture for Cooperative Multi-Agent Systems" Ellis Horwood.