MECHANISMS: A NEW APPROACH TO ROBOTS TEAMS COOPERATION

Silvia S. C. Botelho

FURG *Av. Italia Km8, 96201-000 Rio Grande/RS - Brazil*

Rachid Alami

LAAS-CNRS *7, Avenue du Colonel Roche, 31077 Toulouse Cedex 4 - France*

Abstract— In this paper we propose a new contribution to treat a class of cooperative issues in the multi-robot context. These issues are associated with the common use of some entities, called mechanisms. A mechanism can be seen as a generalization of the notion of resource. The robots can modify its state directly or through requests. The robots can also share its utilization. Multi-robot cooperation will be expressed as a distributed decisional process that tends to solve detect and treat resource conflict situations as well as sources of inefficiency. We discuss these issues and illustrate them through a simulated system, which allows a number of autonomous robots to plan and perform cooperatively a set of servicing tasks in a hospital environment.

Key Words— multi-robots, intelligent architecture, multi-agents, task allocation

1 Introduction

Starting from the **Plan-Merging** Paradigm (Alami et al., 1995) for coordinated resource utilization - and the **M+ Negotiation for Task Allocation - M+NTA** (Botelho and Alami, 1999) for distributed task allocation, we have developed a generic architecture for multi-robot cooperation (Botelho and Alami, 2000). This architecture is based on a combination of local individual planning and coordinated decision for incremental plan adaptation to the multi-robot context. In this paper we present the *mechanism* concept, which allows a set of autonomous agents not only to perform their tasks in a coherent and non-conflict manner but also to cooperatively enhance their task achievement performance. After a brief analysis of related work, we present a general architecture for multi-robot cooperation. We introduce the *mechanisms* and focus on the cooperative plan enhancement issues. Finally, we present an implemented system which illustrates, in simulation, the key aspects of our contribution.

1.1 Related work

The field of multi-robot systems covers today a large spectrum of topics (Parker, 2000). We restrict our analysis here to contributions proposing cooperative schemes at the architectural and/or decisional level. In such stream, *behavior-based* and similar approaches (Mackenzie and Arkin, 1997; Parker, 1998), propose to build sophisticated multi-robot cooperation through the combination of simple (but robust) interaction behaviors.

AI-based cooperative systems have proposed domain independent models for agents interaction. For example, Brafman (Boutilier and R., 1997) enriches the STRIPS formalism, aiming to build centralized conflict-free plans. Several generic approaches have been proposed concerning goal decomposition, task allocation and negotiation (DesJardins et al., 1999). PGP (Decker and Lesser, 1992)) is a specialized mission representation that allows exchanges of plans among the agents. Cooperation has also been treated through negotiation strategies (Rosenschein and Zlotkin, 1994) like CNP-based protocols (Smith, 1980), or BDI approaches where agents interaction is based on their commitment to achieve individual/collective goals (Jennings, 1995; Tambe, 1998). Cooperation for achieving independent goals has been mostly addressed in the framework of application-specific techniques such as multi-robot cooperative navigation (Brumitt, 1996; Azarm and Schmidt, 1997).

1.2 Cooperation for Plan Enhancement

In the context of autonomous multi-robot systems, we identify three main steps that can often be treated separately: the *decomposition* of a mission into tasks (mission planning), the *allocation* of tasks among the available robots and the *tasks achievement* in a multi-robot context (Fig. 1). In this paper, we limit ourselves to this last aspect i.e. the concurrent achievement of a set of tasks. Indeed, we assume a set of autonomous robots which have been given a set of partially ordered tasks. This could be the output of a central planner, or the result of a collaborative planning and task allocation process (Botelho and Alami, 1999). One can consider this plan elaboration process is finished when the obtained tasks have a sufficient range and are sufficiently independent to allow a substantial "selfish" robot activity.

However, and this is a key aspect in robotics, the allocated tasks cannot be directly "executed" but require further refinement taking into account the execution context. Since each robot synthesizes its own detailed plan, we identify two classes

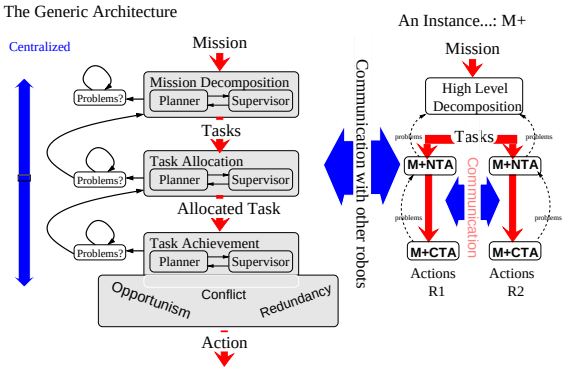

Figura 1. Our architecture for multi-robot cooperation

of problems related to the distributed nature of the system: 1. coordination to avoid and/or solve conflicts and 2. cooperation to enhance the efficiency of the system. The first class has been often treated in the literature. The second class is newer and raises some interesting cooperative issues linked to the improvement of the global performance by detecting possible enhancements. We have developed a distributed cooperative scheme (Botelho and Alami, 2000) called *M+ cooperative task achievement* - `M+CTA`. It is based on the *mechanism* concept and provides a framework for dealing with issues such as opportunistic action re-allocation, detection and suppression of redundancy and incremental/additive actions accomplished by several robots.

2 M+CTA & *mechanism*

In `M+CTA` the task achievement level is based on an incremental plan validation process. Starting from a task that has been allocated to it, a robot, R_p, plans its own sequence of actions, called *individual plan*. This plan is produced without taking into account the other robots' plan. After this planning step, R_p negotiates with the other robots in order to incrementally adapt its plan in the multi-robot context.

A number of conflict/cooperative situation problems are raised when a group of agents share the common use of some entities or devices in the environment. The *mechanisms* provide a suitable framework for robot cooperation. Indeed, there are numerous applications and particularly for servicing tasks, where the robots often need to operate or to interact with automatic machines or passive devices in order to reach their goals or to satisfy some intermediate sub-goals that allow them to finally reach their main goals. For example, a robot has to open a door in order to enter a room, or heat the oven to a given temperature before cooking a cake, etc.. The *mechanism* can be seen as an extension of the concept of resource token: a robot not only allocates and frees a mechanism, it not only consumes or produces it, it can also explicitly manipulate it or act on it, directly or through requests to a controller attached to the mechanism.

The *mechanisms* will allow: 1. to identify the entities of common use, 2. to fix rules to guarantee correct and coherent cooperative *utilization* of such entities and 3. to negotiate their common use among the agents.

2.1 A scenario of cooperation

A *mechanism* is a data structure that defines how to use a device or a machine (the possible sequences of operations, in what conditions it can be shared or used simultaneously by several users, etc). In the current version of our system, this knowledge is represented by (see figure 2): 1. known initial and final states, 2. a set of alternative *paths* (each path is partially instantiated and represents a valid sequence of actions and state changes of the associated entity) and 3. a set of *social rules*.

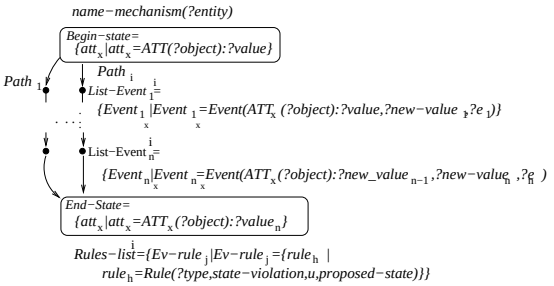

Figura 2. A generic mechanism M

Social Rules impose constraints that must be taken into account during the *mechanisms* use. They have been introduced in order to allow the robots produce easily *merge-able* plans. Social rules are always associated with some mechanism states, which, in particular situations, are not allowed. Social rules specify forbidden or undesirable states and propose states that satisfy the rules. This field is used by the planner in order to avoid the violation of the rule. Social rules are domain dependent; the current version of our system deals with three types of constraints:

1. `amount`: where the "resource" $violation\_state = (att(?object) : v)$ represented by an attribute att and a value v is limited to a maximum number of s agents. Note that such rules allow to describe the resource constraints of the system. For instance *a limitation of 2 robots at desk D1* can be represented by RULE(`amount`,$pos-robot(?r)$,2,$D1$,`OPEN_AREA`), where it is proposed to send the robot to an `OPEN_AREA`, in order to satisfy the rule.

2. `end`: where *proposed_state* must be satisfied at the end of each robot utilization of the resource. This class guarantees a known final state, allowing the planner to predict the state of an attribute (initial state for the next plan).

3. `time`: where *violated_state* can be maintained true only during a given amount of time s.

The use of social rules in the planning phase: We associate to the social rules a scalar value called *obligation level*. Whenever a robot plans, it considers the *proposed states* of the rules as mandatory goals that will be added to its current list of goals. However, depending on the obligation level, goals will be posted 1. as a conjunction with the current robot goals or 2. as additional goals that the robot will try to satisfy in subsequent planning steps. In such case, the planner will produce *additional plans* that will achieve each low-level obligation social rule.

During the execution of a plan, the robot may or may not execute these additional plans, thus neglecting temporarily the *proposed state*. Note that if another agent asks the robot to fulfill the *rule proposed state*, it will then (and only then) perform the associated additional plan. The *obligation level* may also change depending on the context[a].

2.2 Mechanisms & Jobs

Whenever a robot R_p detects that its plan uses an entity associated with a mechanism M, it builds a *job* M_j^p. A *job* is a dynamic structure, which results from the instantiation of a *path* of a given mechanism by the current robot plan. A job is composed of *steps*. Each *step* has a set of information associated with it: for instance, the agent that effectively executes the action, the other plan actions that depend on it (`successors`), etc. *Jobs* are used as structure and language of negotiation allowing R_p and other agents to decide about the common utilization of an entity. Figure 3 shows a plan produced the robot R_p that uses a furnace. R_p builds a job M_j^p and that will be negotiated. This *job* ends when the final state of the associated mechanism is reached.

3 Cooperation with *mechanisms*

The `M+CTA` level involves three activities that correspond to different temporal horizons and may run in parallel: 1. task planning which produces an individual robot plan; 2. the plan negotiation activity which adapts the plan to the multi-robot context; and 3. the effective plan execution.

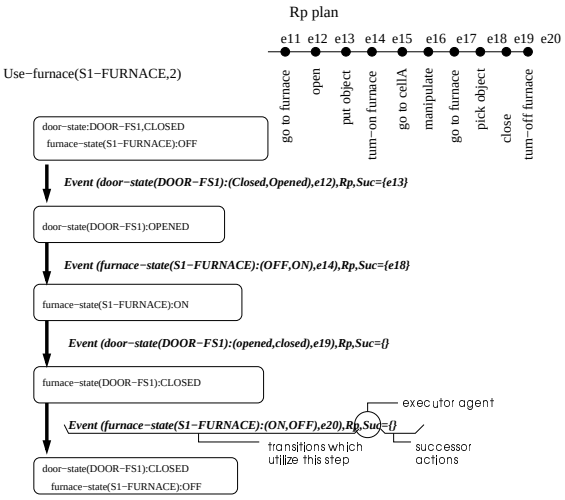

Figura 3. A *job* corresponding to the use of a furnace by *Rp*

From time to time, depending on higher level requirements, the robot invokes its own planner and it incrementally appends new sequences of actions to its current individual plan. This is a standard task planning activity; however, the obtained plan satisfies the social rules and is consequently easily *merge-able*.

Incremental plan negotiation Let us assume that R_p has an individual plan composed of a set of actions A_i^p which manipulate mechanisms. It performs an incremental negotiation process in order to introduce each action A_i^p in the multi-robots context. This operation is "protected" by a mutual exclusion mechanism[b]. The result is a coherent plan which includes all the necessary coordinations and some cooperative actions.

The negotiation process comprises the **announcement** and the **deliberation**. During the announcement process, whenever a robot,R_p needs to validate an action A_i^p (belonging to *job* M_j^p, corresponding to the use of a mechanism M). It announces its job, obtaining current list of jobs involving M. Having the current job list, R_p needs to deliberate, it has two alternatives associated with its job M_j^p and each member list M_j^q, see figure 4:

Fusion: since our robots are cooperative, the aim is to enhance as much as possible the overall performance. Thus, the robot always try to merge his *job* with the current (already negotiated) *jobs* M_j^q. This is done by trying to detect and suppress redundant transitions. The result is a new job

[a]Note that this notion of social rules is different, or even complementary, from the social behaviors proposed by (Shoham and Tennenholtz, 1995). While *social behaviors* are explicitly coded in its reactive task execution, the *social rules* are used at the robot decision level as constraints in its planning, negotiation and execution activities.

[b]We assume that the robots are equipped with a reliable inter-robot communication device.

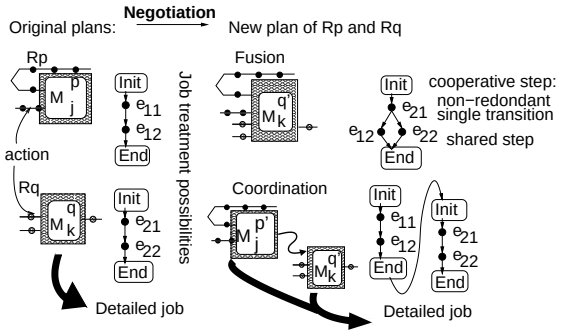

Figura 4. Job treatment possibilities: fusion or coordination.

M_j^q! whose actions may be distributed between the different robots.

However, the constraints imposed by *social rules* may prevent a fusion between two jobs. The only remaining solution is to coordinate them in order to avoid conflicts. *Coordination:* in this situation R_p can use a mechanism M only after its released by the agents associated with M_j^q. In other words, M_j^p has to be coordinated with M_j^q by adding temporal constraints to the jobs.

After each deliberation process, the robots to adapt their plans to the job modification. Note that such a negotiation process involves only communication and computation and concerns future (short term) robot actions. It can run in parallel with execution of the current coordination plan.

Job execution process: Before executing an action A_i^p, the robot **validates** the transition associated to A_i^p. Indeed, a transition remains "negotiable" until its **validation**. Once validated, it is "frozen" and the other robots can only perform insertions after a validated transition. Action execution causes the evolution of the system, resulting in events that will entail new planning, negotiation and execution steps for the robot itself and for the other robots.

4 Illustration and future work

We have implemented a first version of the overall system and run it on simulation. We describe here below some of the obtained results. The application domain that we have chosen is a set of mobile robots in a hospital environment. Servicing tasks are items delivery to beds as well as bed cleaning and room preparation[c]. Fig. 5 shows the simulated environment and 14 partially ordered tasks: `T0,...T13` and the initial world state description. Each robot is equipped with a STRIPS-based task planner and a motion planner.

[c]Each robot control system runs on an independent Sun workstation which communicates with the other workstations through TCP/IP.

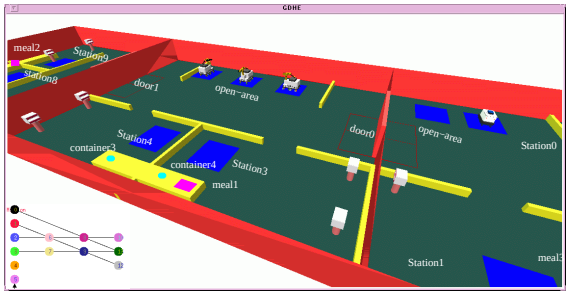

Figura 5. Example 1: Transfer object and clean beds in a hospital area

The robots must negotiate the use of the following *mechanisms*, see figure 6: 1. `clean-room` that allows cleaning actions with cumulative effects when executed several times or by several robots; 2) door-manipulation with `open/close` actions, which can be potentially redundant; and 3) a mechanism that controls the use the dock station by the robots. This mechanism has an `amount` rule (with low *obligation level*) that limits the number of robots near a station to one.

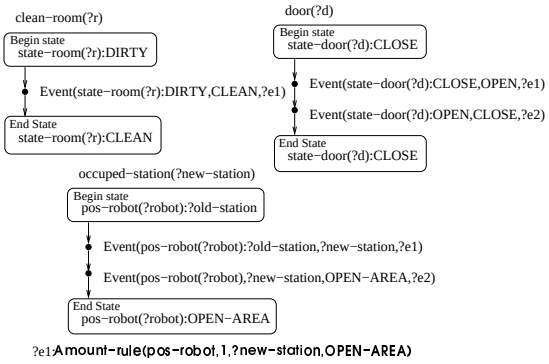

Figura 6. *Mechanisms* to negotiate.

The set of tasks is transmitted to five robots. After a first phase (not described here (Botelho and Alami, 1999)), they plan and incrementally allocate the tasks using *M+ Cooperative Task Allocation*. Fig. 7 shows the individual plans after a number of negotiation processes. Note that `r0` has allocated `T3` in a first step. However it has lost it because `r1` has found a better cost to achieve it. `r1` is achieving `T3`. It has elaborated a plan with six actions in order to achieve its main goal `state-room(S1):CLEAN` and to satisfy the social rule requiring `state-door(D0):CLOSED` with a high obligation level. Besides, it has also produced an *additional plan* that satisfies rule 1 (with a low obligation level) by introducing a `go-to(OPEN-AREA)` action. After several *jobs* negotiation processes, `r1` deletes its `open` action, which will be accomplished by `r3`. This robot will open a first time the door and after all robots take in advantage of this event. Afterwards, `r1` will

close the door for everybody. We can see also the incremental allocation process: while the robots are achieving their current tasks, they try to allocate their future task, for instance: `r1-T8` and `r2-T7`. The arrows between robot plans illustrate the temporal constraints induced by the coordination between jobs.

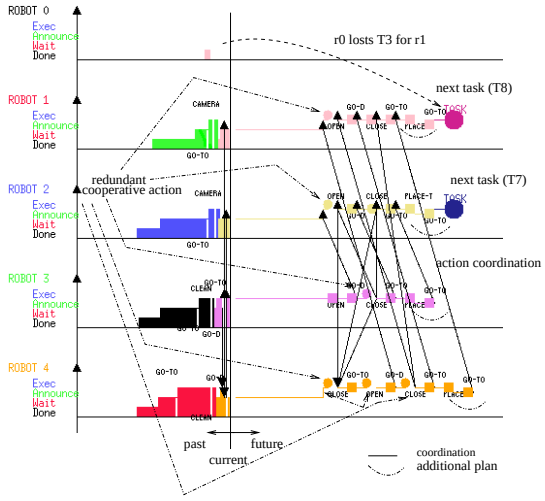

Figura 7. M+ task achievement process.

The overall process continues; the tasks are incrementally planned, negotiated and executed. In the end of this run the robots have satisfied the *social rule* associated to the robot position near the stations. Indeed, some robots delete redundant actions (open/close door), accomplished opportunistically by others. Besides, some robots also helped the others to clean rooms.

Table 8 shows the time sharing among execution and deliberation activities. Deliberation activities are decomposed into task allocation and *mechanisms* negotiation. All activities run in parallel. Note that execution activities are more expensive, however `r0` has a high task allocation activity due to the mission nature and to its proper context: the tasks order limits their execution in parallel and `r0` spends a lot of time searching for a task to perform.

We have run the system several times with different parameter values. These parameters are associated with two aspects: the type of cooperation and the number of robots. We have run the system with three different cooperation strategies: 1. `COOP-TOTAL`: treating redundancy and opportunistic incremental help between *jobs* ; 2. `NO-INC`: only treating redundant cases with no incremental help; and 3. `NO-COOP`: the system allows only coordination between *jobs*.

On the whole, `COOP-TOTAL` enhance the system performance: better cost and less actions (see Figs; 9 and 10).

The number of robots vs. the workload is presented in table 11. We can see that when we have

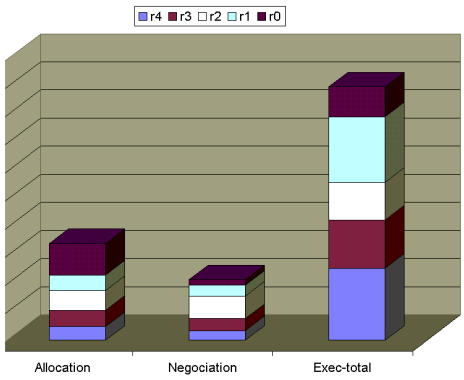

Figura 8. Time Sharing.

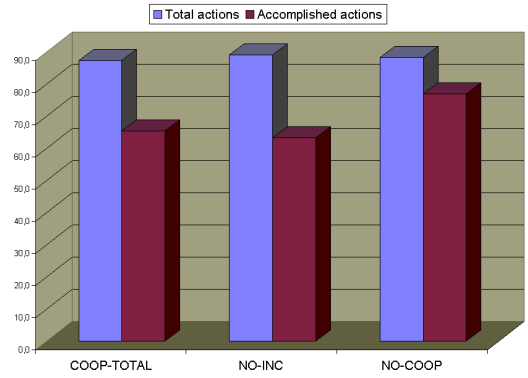

Figura 9. Planned and achieved action by the robots.

5 robots, one of them (`r0`) is almost idle. This fact is due to the nature of mission. The partial order of tasks prevents an optimum deployment of more than 3 robots.However, note that our system has found a very good balance when only three robots are involved.

5 Conclusion

We have proposed and discussed a scheme for cooperative multi-robot task achievement based on *mechanism*. This scheme is a key component of a

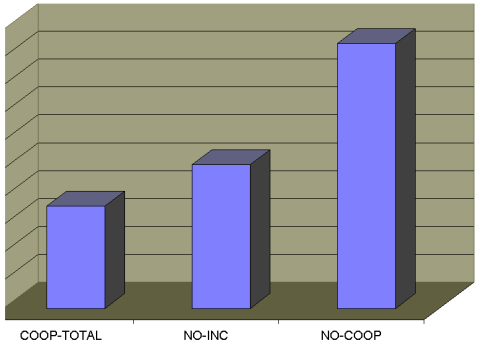

Figura 10. The costs.

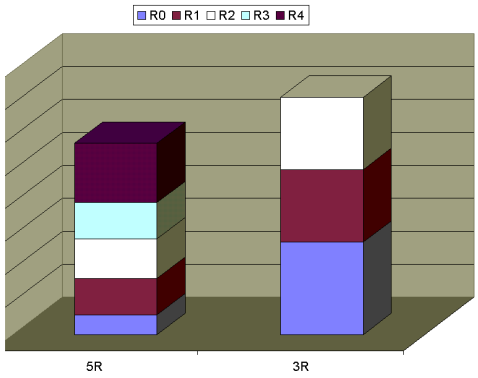

Figura 11. Workload for each robot.

general architecture for multi-robot cooperation. Its main originality comes from its ability to allow the robots to detect and treat - in a distributed and cooperative manner - resource conflict situations as well as sources of inefficiency among the robots. We have presented its main ingredients and operators, and illustrated its use through a simulated system.

We intend to validate our approach through a number of significant different application domains. Besides, we would like to extend and further formalize the overall system and its representational and algorithmic ingredients, taking into account cost and time issues to help planning and negotiation activities.

Referências

Alami, R., Robert, F., Ingrand, F. and Suzuki, S. (1995). Multi-robot cooperation through incremental plan-merging, *IEEE ICRA'95*.

Azarm, K. and Schmidt, G. (1997). A decentralized approach for the conflict-free motion of multiple mobile robots, *Advanced Robotics* **11**(4): 323–340.

Botelho, S. S. C. and Alami, R. (1999). M+: a scheme for multi-robot cooperation through negotiated task allocation and achievement, *IEEE ICRA'99*.

Botelho, S. S. C. and Alami, R. (2000). Robots that cooperatively enhance their plans, *DARS 4, Lynne E. Parker, George Bekey, and Jacob Barhen (eds.), Springer*.

Boutilier, C. and R., B. (1997). Planning with concurrent interaction actions, *AAAI'97*.

Brumitt, B. Stentz, A. (1996). Dynamic mission planning for multiple mobile robots, *IEEE ICRA'96*.

Decker, K. and Lesser, V. (1992). Generalizing the partial global planning algorithm, *Int Journal of Cooperative Information Systems 92*.

DesJardins, M., Durfee, E., C., O. and Wolverton, M. (1999). A survey of research in distributed, continual planning, *AI Magazine* pp. 13–22.

Jennings, N. (1995). Controlling cooperative problem solving in industrial multi-agent systems using joint intentions, *Artificial Intelligence* **75**.

Mackenzie, D. and Arkin, R. (1997). Multiagent mission and execution, *Autonomous Robots* **4**: 29–52.

Parker, L. (1998). Alliance: An architecture for fault tolerant multirobot cooperation, *IEEE Trans. on Robotics and Automation* **14**(2): 220–239.

Parker, L. (2000). Current state of the art in distributed robot systems, *DARS 4, Lynne E. Parker, George Bekey, and Jacob Barhen (eds.), Springer*, pp. 3–12.

Rosenschein, J. S. and Zlotkin, G. (1994). Rules of and encounter: Designing convention for automated negotiation among computers, *Artificial Intelligence - MIT press* .

Shoham, Y. and Tennenholtz, M. (1995). On social laws for artificial agent societies: off-line design, *Artificial Intelligence* **0**(75): 231–252.

Smith, R. (1980). The contract net protocol: High-level communication and control in a distributed problem solver, *IEEE Transactions on Computers* **c-29**(12).

Tambe, M. (1998). Agent architectures for flexible, practical teamwork, *I ICAA*.