Multistage Negotiation For Distributed Scheduling of Resource-Bounded Agents*

Abdel-Illah Mouaddib
CRIL-IUT de Lens-Université d'Artois
Rue de l'université S.P. 16, 62307 Lens Cedex France
+33-3-21-79-32-73 Fax 3-21-79-32-60
E-mail: mouaddib@cril.univ-artois.fr

Abstract

Progressive reasoning was introduced as another way of flexible computation to handle transient overload (resource-bounded) of real-time systems. In a system based on this technique, each time constrained agent is designed in such a way that it can adapt the quality of its solution to the available time of the whole system. The monitoring complements this system with a mechanism that determines for each agent its runtime. We address in this paper how to use Distributed Artificial Intelligence (DAI) methods for controlling self-interested and time-constrained agents where a central controller do not exist. Distributed scheduling consists of each agent locally allocates its time to build its local schedule and then it coordinates its local schedule with those of the other agents. We propose Partial Global Flexible Scheduling (PGFS) which consists of cooperation allowing a new agent to construct its local schedule and negotiation allowing the constructed local schedule to be coordinated with local schedules of the other agents. This approach is an important application of the contract-net technique to the distributed scheduling of resource-bounded and self-interested agents.

1 Introduction

Progressive reasoning (Mouaddib 1993; Mouaddib, Charpillet, & Haton 1992) was introduced as another way of flexible computation to handle transient overload of real-time systems. Agents using progressive reasoning offer a tradeoff between solution quality and computation time. In a system based on this technique, each time constrained agent is designed in such a way that it can adapt the quality of its solution through a hierarchy of reasoning levels (Mouaddib 1993; Mouaddib & Zilberstein 1995), to the available time of the whole system. The control gives a system the mechanism to allocate time to each progressive reasoning unit in order to optimize the performance of the complete system. This problem is studied in different approaches of flexible computation such as *Monitoring anytime algorithms* (Zilberstein 1995), *Design-To-Time* (Garvey & Lesser 1993), *Imprecise Computation* (Liu *et al.* 1991) and *Incremental Scheduling* (Mouaddib & Zilberstein 1997). All these approaches describe centralized mechanism showing how the system allocates time to each one of its components. These approaches assume that central controller exists. This assumption is not valid in several applications such as the airport ground service scheduling (Neiman, Hildum, & Lesser 1994), resource allocation in a communication network (Sugawara 1990), cooperating robots (Mouaddib 1997), air-traffic (Buteau & Brandon 1990) and similar problems may arise in factory floor manufacturing domains. Such applications require coordinated system where agents are able to run distributed scheduling.

What we want to present in this paper is how Distributed Artificial Intelligence methods can be used for controlling self-interested and time-constrained agents where central controller don't exists. In such systems the control can be seen as an activity that it could be distributed among agents. Indeed, we would like that agents locally decide for their local schedules and coordinate their respective local schedules by negotiation (Davis & Smith 1983; Kraus, Wilkenfeld, & Zlotkin 1995). This negotiation arises when local schedulers must borrow resources (time) from other local schedulers in order to resolve local impasses or improve the quality of the local schedules.

The scenario we consider consists of self-interested and time-constrained agents that are able to progressively determine their local plan (set of actions), to schedule it and to react properly to local schedules of other agents. We assume that agents have to solve a problem before a given deadline. An agent organizes its solution in an incremental way through a hierarchy of reasoning levels where each of which returns a solution with quality better than the one of the higher

This work has been supported by the Ganymède-II project of the Contract Plan Etat/Nord-Pas-De-Calais, by the MENESR and l'Institut Universitaire de Technologie de Lens

levels (Mouaddib & Zilberstein 1995). Then, the local schedule consists of a window of time at which all levels must be executed. Conflicts between local schedules occur when sorting local schedules, since some deadlines could be violated. The strategy that we study consists of coordinating different local schedules to avoid conflicts by minimizing the backtracking (rescheduling). We propose the Partial Global Flexible Scheduling (PGFS) approach which allows cooperation between agents to gather together temporal relations between them in order to build local schedules, and incremental negotiation (Mouaddib 1997) in order to coordinate their local schedules.

In the following, section 2 presents a short example on which this approach is assessed. Section 3 describes the distributed multi-agent scheduling with which we work. Section 4 describes our approach for a PGFS coordination approach. Section 5 presents the comparaison to related work. The paper concludes in section 6 and gives the future work.

2 Short Example: Coordinating Robots

Our approach is under assessment in a simulated multiple robot environment. In this environment, robots evolve in the same area where they load goods from one location and they unload them in another location. The robots visit the "battery loading" room at a particular period of time. The robot constructs a schedule for visiting the room to load its batteries (3 batteries for example). The conflicts arise when schedules overlap. We do not address, in this paper, the problem of the coordination of robots trajectory that it is studied in another work. The progressive processing of robots consists in loading progressively the battery one by one where each processing level is the loading one battery task (constructing the path to the appropriate power distributer in the room and loading the battery). The robots are self-interested and try to maximize their own utility by avoiding to visit several time the "battery loading" room. For this, each robot constructs a schedule for the processing levels (loading all batteries). The distributed nature of the application do not allow a centralized control for anticipating and solving conflicts. Consequently, when conflicts occur, robots negotiate by reducing their time intervals until a consistent state is reached. This means that robots flex their constraints and they tolerate not loading their least important battery (the ones used for tasks other than moving).

The robot, from its percept that give the state of batteries, computes the deadline at which the batteries must be charged (the most important battery at least). Through the multistage CNP we propose, the robot declares its intention to insert its activity (loading batteries) to the society of already coordinated robots and then it coordinates it with them. This coordination consists of the interval of time w_α (a local schedule) during which it uses the "battery loading" room. The robots can dynamically merge their activities in the robot society.

3 Distributed multi-agent scheduling

The environment with which we are concerned, consists of distributing different problems on agents, each of which solves a problem (Jennings 1995). This kind of environments concerns problems where a global goal is decomposed into subgoals and "Divide and conquer" is the main technique used to reduce the complexity of the problems (Ephrati & Rosenschein 1994).

Our scenario consists of a group $A = \{\alpha, \beta, ..., \gamma\}$ of n agents. These agents are to achieve a set of problems $\{ G_1, G_2, ..., G_n \}$. Each problem G_i is solved at varying levels of details through a hierarchy of reasoning levels $\{ L_i^1, L_i^2, \ldots, L_i^n \}$ by the agent i before the deadline D_i. In the multi-agent scenario we are working with, each agent proposes the optimal subset of its reasoning level set that optimizes the solution quality in the available time. The agent sees only the goal it must achieve and the deadline before which its execution must be finished. To define the number of levels it must execute, the agent must cooperate to define its temporal location according to the temporal location of the other agents in the society. The result of this cooperation must allows the agent to define the execution's *earliest start time* while its execution's *latest end time* is its start time plus the amount of time required to execute all reasoning levels. Negotiation is processed when its end time is greater than its deadline. This negotiation borrows resources (time) from the least important agents in the society.

To summarize, our approach allows distributing among agents the main steps of scheduling algorithms: first, *temporal location* that allows inserting new agent in the current global schedule. Second, *flexing local schedules* that allows borrowing resources (discarding reasoning levels) from the least important agents when the insertion of the new agent fails. Third, *propagating the local schedule of the inserted agent* that allows globally scheduling the local schedule with the other of agents and to verify that no deadline is violated. Consequently, the PGFS is designed in such a way that it proceeds on a distributed search among agents to collect a distributed temporal relations for building a local schedule followed by a distributed search among agents to collect conflicts between local schedules.

4 Partial Global Flexible Scheduling approach

The approach PGFS that we propose is based: a *cooperation step* allowing the merging agent to gather together temporal information to build its local schedule and *a negotiation step* between the merging agent and agents in the society to coordinate their local schedules.

4.1 Progressive reasoning agents

A progressive reasoning agent consists of a composite unit α of reasoning levels L_α^i organized on a linear precedence-constraint graph. The level L_α^i can begin execution only after the level L_α^{i-1} completes. The level L_α^i is the immediate successor of L_α^{i-1}, and the output of L_α^{i-1} is one of the inputs of L_α^i. The output of the last level $L_\alpha^{n_\alpha}$ is the output of the unit α. The first level L_α^1 is named the *mandatory level* while the other levels are named *optional levels.*

The agent α must optimize the time/quality tradeoff by allocating time to each level within the available time. What we address in this paper is how to optimize the time/quality tradeoff of the agent society by using cooperation and negotiation techniques.

4.2 Agent interactions

Agents communicate to share their points of view in order to anticipate any conflicts between them. The communication is based on a message-passing mechanism (Mouaddib 1993). An agent needs to communicate in several situations: (1) the agent broadcasts its deadline to be merged in the society, (2) the merging agent receives temporal relations (before, after) that link it with the agents of the society, (3) the merging agent constructs its local schedule and sends a disagreement to borrow resources (time) from other agents when its schedule fails, (4) the merging agent receives from the other agents of society the resources it can tolerate to borrow, (5) the merging agent uses those resources to build a succeful local schedule that it sends to the other agents in society, (6) the merging agent receives disagreement from the other agents in society because their respective deadlines are violated, (7) the merging agent becomes an agent of the society and elects new merging agent from the community of agents sending disagreement and then another cycle of multistage of CNP starts. These steps are achieved through successive CNPs that we describe in the following.

4.3 Cooperative construction of the local schedule

Cooperation between agents is necessary because no agent has a sufficient global view on temporal relations linking them, and negotiation is necessary to resolve local impasses. Indeed, the merging agent cooperates to collect temporal relations linking it to the other agents, then it constructs its local schedule. The constructed local schedule consists in determining the window w_α during which its reasoning levels could be executed. This window is characterized by its *Begin-Time* (BT) and *End-Time* (ET). We describe in this section the cooperation between agents to construct a local schedule for a merging agent. We denote as α, the new agent who tries to merge its local schedule in the society. The cooperative construction of the local schedule of the new agent (that we call also the merging agent) is based on the following steps:

- *Broadcasting the deadline*: The agent α broadcasts its deadline D_α (announcement) in order to collect the temporal relations {*before, after*} (bids) that allow it to be located before/after another agent when its deadline is less/greater than the one of the other. In the example shown in the figure 1, the agent M (merging agent) broadcasts its deadline 9 to the agents { A, B, C, D } of the society.

- *Sending temporal relations*: Each agent β of the society receives a deadline D_α of α and compares it to its deadline. The agent β sends the *after/before* temporal relation when its deadline is greater/less than D_α. In the example of the figure 1, the agents A and B send *before* temporal relation while C and D send *after* temporal relation.

- *Organizing society as communities*: The agent α receives from each agent the temporal relation that links it to this agent. The agent α thus organizes the society as two communities: *List-of-Before* and *List-of-After* that contain respectively agents linked with *before* and agents linked with *after* to the agent α. As shown in the figure 1, agents A and B are in *List-of-Before* community while C and D are in *List-of-After* community.

- *Determining the local schedule*: Once the two communities are identified, the agent α starts the construction of the local schedule. This latter is defined by a search among *List-of-Before* to find the latest end time (LET). This latter is the end time of the nearest agent (nearest deadline to the deadline of the merging agent) in the *List-of-Before.* The nearest agent is found from the intersection of different *List-of-After* of agents in the *List-of-Before* of the merging agent. Thus, The merging agent defines its BT as LET while its ET is BT plus the amount of time necessary to execute all the reasoning levels of the merging agent. The merging agent tries to schedule all its reasoning levels in order to maximize its own utility. Formally, $w_\alpha = [BT_\alpha, ET_\alpha]$ where $BT_\alpha =$

$ET(\bigcap_{\beta \in List-of-Before}(List-of-After(\beta)))$ and $ET_\alpha = BT_\alpha + \sum_{i=1}^{i=n_\alpha} Computation\_time(L_\alpha^i)$
In the figure 1, the begin time of the merging agent

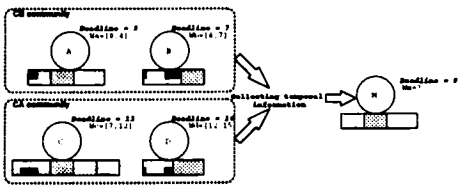

Figure 1: Cycle of collaborative construction

M, BT_M, is the end time of the agent B (nearest agent to M) which is equal to 7. Thus $BT_M = 7$. Let the computation times of the three levels of M be {1, 1, 2}, thus the end time of the agent M is $ET_M = 11$.

The local schedule constructed is with maximal utility. The merging agent broadcasts a disagreement to the *List-of-Before* community when its ET_α is greater than its deadline D_α. This is the case in our example ($ET_M = 11 > D_M = 9$). This disagreement consists in asking the agents in the *List-of-Before* community to flex their local schedules to borrow resources (time) in order to maintain its maximal utility. We discuss this phase of negotiation in the next section.

4.4 Negotiation for flexing local schedules

The merging agent announces to the agents in *List-of-Before* that its local schedule allowing maximal utility is out of reach (deadline violated). This announcement conveys the amount of time by which the local schedule violates the local deadline. In the example of figure 1, the agent M sends a disagreement to the agents $\{A, B\}$ conveying the amount 2 by which ET_M exceeds D_M. This amount of time allows each agent in the *List-of-Before* to determine the number of reasoning levels it can tolerate discarding. In the example of figure 1, the agent A could tolerate discarding its second and third levels where their respective computation times are {1, 2} while the agent B could discard its second level when its computation time is {1}. The merging agent receives from each agent β in the *List-of-Before* a list LLD_β (bids) that contains reasoning levels tolerated to be discarded and their computation time. The negotiation between the merging agent and the agents in *List-of-Before* is based on the following steps:

- *Announcement:* the agent α broadcasts the amount of time to borrow AT by which the merging agent violates its deadline $AT = ET_\alpha - D_\alpha$ to *List-of-Before*. In our example of the figure 1, AT = 2 is broadcasted to {A, B}.
- *Partial Global Bidding*: Proposing the lists LLD_β consists of once an agent β receives AT, it searches in its set of optional reasoning levels the optimal subset necessary to fulfill AT. For this end, the agent β selects the lowest levels in its hierarchy as tolerated discarding levels until the AT is fulfill or all optional levels are selected (partial bid). In our example, the agent A proposes $LLD_A = \{L_A^3\}$ because its computation time allows AT to be fulfilled while the agent B proposes $LLD_B = \{L_B^2\}$ although the AT is not fulfilled but it is the most tolerated list.
- *Partial Global Awarding*: The merging agent α receives the lists LLD_β (bids) to optimally determine for each agent β the reasoning levels to discard including those in its own hierarchy. For this end, the merging agent uses ***utility function*** of reasoning level. Indeed, the strategy of the merging agent prefers to borrow time from agents having levels with the least utility. The merging agent sorts (in increasing order) the levels of all the lists LLD including its own regardless of their utility. Afterwards, it selects levels one by one until the AT is fulfilled or all levels are selected. The merging agent sends, thus, for each agent β its selected levels (partial award) to discard and consequently reduces its local schedule. Let in our example L_A^3 be the level with the least utility. Thus, the agent M sends to the agent A that this level is selected to be discarded. In our example, this is enough to fulfill AT.
- *Taking awards*: Each agent β receives the lists of reasoning levels selected to be discarded (award), LSD. Thus, the agent β reduces its window w_β by the amount of time $RAT = \sum_{L \in LSD} C_L$. Formally, the window w_β becomes: $[BT_\beta, ET_\beta - RAT]$. In the example, the window of the agent A becomes [0,2]. Afterwards, the agent β broadcasts the amount of time gained RAT after discarding its reasoning levels to the agents η in its *List-of-After*. In the example, the agent A sends to { B, M, C, D } the amount of gained time $RAT = 2$.

  The agents η in *List-of-After* of the flexed agent's local schedule receives the amount of time RAT by which they must update their windows w_η. This update consists in moving the windows w_η as follows $[BT_\eta - RAT, ET_\eta - RAT]$. In the example, the windows of B, M, C and D become [2,5]; [5,9]; [5,10]; [10, 13].

The local schedule of the merging agent must be feasible after this negotiation. Otherwise, the agents in *List-of-Before* could not flex their local schedules more that they have been proposed (discarding optional levels). Consequently, some of the mandatory reasoning

levels (with the least utility) need to be discarded and thus the local schedules of their corresponding agents are refused. We see that the CNP proposed here allows to satisfy announcement by a subset of bids received but not only with one bid unlike to the classical CNP.

4.5 Negotiation for merging local schedule

The negotiation between the merging agent α and the agents in the *List-of-Before* allows the merging agent to have a consistent view on the local schedules. Once the consistent view is established, the merging agent broadcasts its local schedule to the agents in the *List-of-After* to verify the consistency on their local view. The consistency of a local view represents the fact that when an agent η in the *List-of-After* receives the local schedule of the merging agent and moves its window w_η with the duration of the window w_α, its deadline D_η remains respected. In contrast, when the deadline D_η is violated, the agent η sends a disagreement and it becomes a candidate to be "new merging agent". We describe next the negotiation with the *List-of-After*:

- *Announcement*: Broadcasting the merging local schedule conveying its window w_α to the agents in the *List-of-After*. In the example, the agent M sends [5,9] to {C, D}.
- *Bidding:* On receipt of the merging local schedule, agents η (of the *List-of-After*) move their local schedule with the duration dt_α of the window w_α. Formally, $w_\eta = [BT_\eta + dt_\alpha, ET_\eta + dt_\alpha]$. In the example, the agents C and D move their windows [5,10] and [10, 13] with $dt_M = 4$. Thus, the windows of C and D become respectively [9,14] and [14, 17]. When the agents η move their windows w_η, they formulate a disagreement (bid) if their deadline is violated (ET_η becomes greater than D_η). In our example both C and D send a disagreement.
- *Awarding*: The merging agent α receives disagreements (bids) from the agents η and have to elect the new meging agent. Each agent η needs local schedules of agents in its *List-of-Before* to be flexed. To avoid redundant processing, our strategy chooses the *List-of-Before* common to all agents η. To this end, we select the *List-of-Before* of the agent η with the *earliest local schedule*, that is with the earliest deadline. In the example, C is elected a new merging agent while the agent M is merged in the society. The new merging agent repeats the "strategy of flexing local schedules" step followed by the "propagation of merging local schedule" step. This processing is repeated until no disagreement is received by the merging agent.

5 Comparaison to related work

It is trivially true that a centralized approach will take longer than our approach because of the parallelism aspect of our solution. The computational cost in our approach has a formulae such as $MAX_{\beta \in society}(Interaction\_cost(\alpha, \beta))$ while this computational cost in centralized approach has a formulae as $\sum_{\beta \in society} Interaction\_cost(\alpha, \beta)$. Further, the number of negotiation rounds is bounded by the number n of the agents in the society. This guarantees that PGFS always terminates.

Distributed scheduling is an important problem that has been addressed by many other investigators such as TRACONET (Sandholm 1993), MARS (Fischer, Muller, & Pischal 1995), CP/CR (Liu & Sycara 1994) and (Sen & Durfee 1996). Unlike these approaches, the approach we present allows a new agent to cooperate for constructing its local schedule and they negotiate to solve conflicts between them. The main differences with TRACONET (and MARS) are: first TRACONET is a task negotiation protocol while our approach is resource negotiation protocol. Second, TRACONET assumes announcement constructed in a local decision while in our approach is a copperative construction result. Third, TRACONET assumes that agents are able to send a bid to fulfill the complete announcement while in our approach, agents send bids to fulfill a part of announcement. Consequently, the announcements could be fully fulfill among different bidders (there are several bidders selected but not only one unlike previous approaches) or annoucements could be partially fulfill. While the differnces with CP/CR approach consist of the fact that our approach utilizes incremental coordinating of local schedules to produce complete schedule. Indeed, our approach allows to coordinate a new local schedule with established and coordinated local schedules while CP/CR approach builds an initial schedule that it revises incrementally. This difference shows the fact that at any moment, our approach can propose a complete feasible schedule while CP/CR approach can not. Consequently, our approach is more suitable to dynamic situations.

Furthermore, this approach is an application to distributed scheduling of PGPP approach (Mouaddib 1997) which is an extension of PGP (Durfee & Lesser 1987) to progressive reasoning. We present a conceptual and formal model for distributed construction of the announcement (local schedule), borrowing resources announcement, bidding and awarding decisions where agents locally decide for resources they tolerate to borrow (partial/complete bidding). This approach extends the contract net protocol in different ways. First, announcements are results of co-

operation between agents. Second, contractees propose partial contracts and they fulfill a part of the announcement unlike the work in the field of extended contract net protocol (Schwartz & Kraus 1997; Sandholm & Lesser 1996; Sandholm & Ygge 1997). Then, the PGFS introduces new notions in contract-net protocol such partial bids, partially fulfilled announcements and partial awards.

6 Conclusion and open questions

We have presented in this paper the PGFS approach dedicated to systems of progressive reasoning units that are distributed, self-interested, want to maximize their own performance and there is no central controller. This approach consists of three successive CNPs allowing distributed temporal location, distributed search of conflicts and distributed resolution of them. This approach introduces the bidding step of a CNP which selects a set of bidders but not only one. In the same way, the announcement (contract) can be partial fulfilled. Another difference is that our approach is suitable to dynamic situations in sense that new agent can be merged in society. Furthermore, Our approach can be interrupted at any time and proposes a complete and feasible schedule. Future work concerns: (1) introducing another cycle of negotiation for monitoring the execution and handling execution failure and (2) introducing the notion of believability" representing the degree that an agent tells truth.

References

Buteau, and Brandon. 1990. Generic framework for distributed, cooperating blackboard systems. In *ACM Computer Science Conference*, 358–365.

Davis, R., and Smith, R. 1983. Negotiation as a metaphor for distributed problem solving. *Artificial Intelligence* 20(1):63–101.

Durfee, E., and Lesser, V. 1987. Partial global planning: A coordination framework for distributed hypothesis formation. *IEEE Transactions on systems, Man, and Cybernetics* 21(5).

Ephrati, E., and Rosenschein, J. 1994. Divide and conquer in multi-agent planning. In *AAAI*, 375–380.

Fischer, K.; Muller, J.; and Pischal, M. 1995. Cooperative transportation scheduling: an application domain for dai. *Technical Report RR-95-01.*

Garvey, A., and Lesser, V. 1993. Design-to-time real-time scheduling. *IEEE Transactions on systems, Man, and Cybernetics* 23(6).

Jennings, N. 1995. Controlling cooperative problem solving in industrial multi-agent systems using joint intentions. *Artificial Intelligence* 75:195–240.

Kraus, S.; Wilkenfeld, J.; and Zlotkin, G. 1995. Multiagent negotiation under time constraints. *Artificial Intelligence* 75:297–345.

Liu, J., and Sycara, K. 1994. Distributed problem solving through coordination in a society of agents. In *IEEE Workshop on Distributed Artificial Intelligence.*

Liu, J.; Lin, K.; Shih, W.; Yu, A.; Chung, J.; and Zao, W. 1991. Algorithms for scheduling imprecise computations. *IEEE Transactions on Computer* 24(5):58–68.

Mouaddib, A.-I., and Zilberstein, S. 1995. Knowledge-based anytime computation. In *IJCAI*, 775–781.

Mouaddib, A.-I., and Zilberstein, S. 1997. Handling duration uncertainty in meta-level control for progressive reasoning. In *IJCAI-97*, 1201–1206.

Mouaddib, A.-I.; Charpillet, F.; and Haton, J. 1992. Approximation and progressive reasoning. In *AAAI Workshop on Imprecise Computation*, 166–170.

Mouaddib, A.-I. 1993. Contribution au raisonnement progressif et temps réel dans un univers multi-agents. *PhD, University of Nancy I, (in French).*

Mouaddib, A.-I. 1997. Progressive negotiation for time-constrained autonomous agents. In *First International Conference on Autonomous Agents*, 8–16.

Neiman, D.; Hildum, D.; and Lesser, V. R. 1994. Exploiting meta-level information in a distributed scheduling system. In *AAAI-94.*

Sandholm, T., and Lesser, V. 1996. Advantage of a leveled commitment contracting protocol. In *AAAI*, 126–133.

Sandholm, T., and Ygge, F. 1997. On the gain and losses of speculation in equilibrium markets. In *IJCAI*, 632–639.

Sandholm, T. 1993. An implementation of the contract net protocol based on marginal cost calculations. In *AAAI*, 256–262.

Schwartz, R., and Kraus, S. 1997. Negotiation on data allocation in multi-agents environments. In *AAAI*, 29–35.

Sen, S., and Durfee, E. 1996. A contracting model for flexible distributed scheduling. *Annals of Operating Research* 65:195–222.

Sugawara, T. 1990. Cooperative lan diagnostic and observable expert system. In *IEEE Phoenix Conf. on Comp. and Comm.*, 667–674.

Zilberstein, S. 1995. Optimizing decision quality with contract algorithms. In *IJCAI-95*, 1576–1582.