Resource sharing in multi-agent systems through multi-stage negotiation

Panos Alexopoulos

IMC Research,
Fokidos 47, Athens, 11527,Greece
palexopoulos@imc.com.gr

Abstract. In this paper we try to solve the Temporal Resource Reallocation Problem (TRR-P) in multi-agent systems by having the agents negotiate periods of time during which they can have use of resources. Our work is based on and extends a previous work in which a multi-stage negotiation framework that defines the way agents negotiate over resources and time is defined. The approach in this framework suggests that the negotiating agents use a sequence of stages, each characterized by an increased chance of success and amount of exchanged information. In the same work two negotiation stages are defined each of which solves a specific range of TRR-P instances. In this paper we propose a third negotiation stage which is more sophisticated than the first two and is able to solve a larger range of TRR-P instances.

1 Introduction

Temporal Resource Reallocation Problem (TRR-P) is a problem which arises when agents own resources for specific periods of time and have activities that need to carry out within other specific time periods which might or might not coincide with the first ones. In the latter case, agents have to exchange resources in order to achieve a resource distribution which will allow them to perform their activities within their specified time-windows. For example, an agent x might own a resource r from time 1 to 10 but need it from time 12 to 15. If another agent y owns r at this interval and does not need it, then y can give r to x. If y, on the other hand, needs r from time 12 to 15, then it cannot give r to x unless it manages to reschedule its activity.

In this paper we attempt to solve TRR-P through a multi-stage negotiation process that is described in [1]. In that paper, two negotiation stages are defined each of which consists of a protocol and a policy according to which agents engage themselves into dialogues. The higher a stage is, the higher is the amount of the exchanged information and the likelihood that a dialogue will be successful. In this paper we extend the negotiating framework of [1] by proposing a third negotiation stage which is more complicated compared to the existing stages but manages to solve a larger range of TRR-P instances. The structure of the rest paper is as follows: The next section describes the negotiation framework as suggested in [1] focusing on the agent representation schema and the negotiation language. In section three we describe the two existing stages of the multi-stage negotiation process defined in [1] and we indicate their merits and limitations in solving the TRR-P problem. Finally, in section four we present our proposed third stage and we define the range of problems it can solve. It should be noted that due to space limitations the exact semantics of the stages' policies are omitted.

2 Negotiation Framework

2.1 Agent Representation

According to [1] the knowledge of an agent is represented as a tuple $< B, R, I, D, G >$ where B is the set of the agent's beliefs (domain-dependent beliefs such as information about itself and the other agents as well as domain-independent beliefs such as dialogue policies), R contains the resources the agent owns before the dialogue starts, I is the agent's current intention (i.e. the plan which will enable the agent to achieve its goal, the agent's available resources and the resources which he needs in order to execute the plan), D contains the current dialogue store (i.e. past dialogue performatives) and G is the agent's goal.

Agent intentions are sets of activities. Each activity is represented as a tuple $< a, R_a, D_a, Ts_a, Te_a >$. This representation denotes that an activity a requires a resource R_a, has duration D_a, its earliest start time is Ts_a and its latest end time is Te_a. Furthermore, the agent's knowledge includes a (possibly empty) *concrete schedule* $\widehat{I}$ for activities represented as $< a, t_s, t_e >$ where $t_e - t_s = D_a$, $t_s \geq Ts_a$ and $t_e \leq Te_a$.

The concrete schedule practically indicates the time interval during which the agent has decided to carry out its activity and it is vital for the agent's request generation mechanism. The reason is that it helps the agent determine what resources to ask from the other agents and for how long. The activity schedule, on the other hand, determines the range of the concrete schedules that the agent can adopt. That is useful when the agent reasons about how to react to another agent's request. For example, in the first stage defined in [1] the agent is willing to change its concrete schedule in order to be able to accept an incoming request. Whether this is possible depends on its activity schedule.

2.2 Negotiation Language

The negotiation language of the framework in [1] defines the *primitives (performatives)* that all negotiating agents use. These dialogue performatives are of the form $tell(X, Y, Move, D, T)$ where X and Y are the sending and receiving agents respectively, T is the time of the performative utterance and *Move* is a *dialogue move*. Allowed dialogue moves are:

- $tell(x, y, request(give(r, (T_s, T_e))), D, T)$: Agent x requests a resource r from agent y for the time interval (T_s, T_e).
- $tell(x, y, accept(request(give(r, (T_s, T_e)))), D, T)$: Agent x accepts y's request.
- $tell(x, y, refuse(request(give(r, (T_s, T_e)))), D, T)$: Agent x denies y's request.
- $tell(x, y, promise(r, (T'_s, T'_e), (T_s, T_e))), D, T)$: Agent x promises to agent y that he shall give him r for the interval (T_s, T_e) if he gives him r for the interval (T'_s, T'_e)".
- $tell(x, y, accept(promise(r, (T'_s, T'_e), (T_s, T_e))), D, T))$: Agent x accepts the exchange proposed by y.
- $tell(x, y, change(promise(r, (T'_s, T'_e), (T_s, T_e))), D, T))$: Agent x asks from agent y to propose a deal different that the one already proposed.

Finally, D is the dialogue in which the dialogue move belongs. Each time a dialogue performative is uttered, it is recorded on a blackboard *(dialogue store)* which is shared amongst the agents. This store grows monotonically as a dialogue takes place and it is reset when the dialogue ends.

3 Existing Negotiation Stages

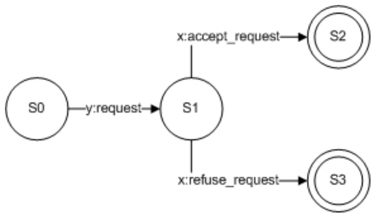
A negotiation stage consists of a **protocol**, namely a set of rules to which the negotiating agents conform and a **policy**, namely a set of rules that the agents use for generating dialogue locutions. It is also characterized by the amount of information that the agents need to exchange in order to reach an agreement. The approach in [1] suggests that the negotiating agents use a sequence of stages, each characterized by an increased chance of success and an increased amount of exchanged information. If a stage fails to produce an agreement, then the agents adopt the next stage in the sequence.

The first negotiation stage defined in [1] conforms to the protocol depicted in figure 1. In this stage, an agent accepts a request for a resource r either if it doesn't intend to use r during the requested interval or if it can change its activity schedule so that it doesn't need r in that interval.

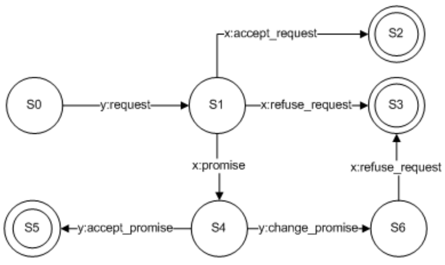
The dialogues that can be generated by conforming to the protocol of figure 1 have a fixed number of steps which means that they always terminate. The range of allocation problems that can be solved by the second stage is defined as follows: if there exists a time window in which an agent x needs a resource r and another agent y has this resource available in that window then the stage's policy can generate a dialogue which solves x's reallocation problem about r.

A case in which Stage 1 fails to solve the reallocation problem is when the initial resource assignment is the following: $I_y = \{< a, r, 5, 10, 20 >\}$, $I_x = \{< b, r, 5, 10, 15 >\}$, $R_y = \{have(r, 10, 15))\}$, $R_x = \{have(r, 15, 20))\}$. It is obvious that y cannot give r to x since it won't be able to carry out activity a.

With such cases deals the second negotiation stage which tries to solve a greater range of problems than the first one by allowing a more elaborate interaction between the agents. This stage's protocol is depicted in figure 2. This stage is different from stage 1 in that the agent that receives a request can



(a) Stage 1 protocol.



(b) Stage 2 protocol.

accept it or deny it or, additionally, propose a deal. The requester can either accept the deal or reject it.

Considering the example that stage 1 could not solve, stage 2 solves it through the following dialogue:

- $tell(x, y, request(give(r, (10, 15))), d(2), 1)$
- $tell(y, x, promise(r, (15, 20), (10, 15))), d(2), 2)$
- $tell(y, x, accept(promise(r, (15, 20), (10, 15)))), d(2), 3)$

As it can be inferred by the protocol of stage 2, dialogues might never terminate. Termination is ensured by making the dialogue terminate after a certain time T_{max} and by not allowing repetition of dialogue moves. The range of problems that are solved by stage 2 is defined by the following theorem:

Theorem: If a system consists of two agents x,y, each having an initial resource assignment, then for all system resources r, all activities a and b , all times τ and all intervals $[T_s, T_e]$ such that $K_{x,\tau} \vdash miss(r, (T_s, T_e), a) \wedge K_{y,\tau} \vdash need(r, (T_s, T_e), b) \wedge \exists [T'_s, T'_e]$ such that $K_{y,\tau} \cup \{give\_away(r, (T_s, T_e)), obtain(r, (T'_s, T'_e))\} \vdash feasible(b, (T''_s, T''_e))$ for some $[T''_s, T''_e]$, there exists a dialogue d, induced by the policy of Stage 2, starting at time τ and ending at time τ' such that $K_{x,\tau'} \vdash need(r, (T_s, T_e), a, \tau') \wedge K_{y,\tau'} \vdash indiff(r, (T_s, T_e))$.

The symbol $K_{x,\tau'}$ denotes the agent x's knowledge at time τ. The predicate **miss** denotes the agent's lack of a required resource in a given interval, **need** that the agent cannot give away a resource for a given interval and **feasible** his ability to perform an activity during a given interval.

The above theorem practically says that if agent x can propose y an alternative time interval during which y can carry out his activity (by x giving y resource r), then y will change its schedule and will give resource r to x. Thus the allocation problem is solved by this exchange of resource r.

However, stage 2 doesn't cover the case in which more than one exchange is required. For example, if the resource and activity assignment is as follows: $I_y = \{< a, r, 2, 10, 15 >, < c, r, 2, 15, 20 >\}$, $I_x = \{< b, r, 5, 10, 20 >\}$, $R_y =$

$\{have(r, 13, 17))\}$, $R_x = \{have(r, 10, 13)), have(r, 17, 20))\}$ then x will try to get r from y either for the period [13,15] or period [15,17]. In the first case y will answer by asking r either for the period [10,12] or [11,13] neither of which makes x's activity b feasible. The same happens if x asks r for period [15,17] so there is no dialogue that can lead to a solution.

4 Stage 3

4.1 General description

Stage 2 cannot solve problems where more than one exchange is required and the reason is that in a dialogue, only one of the agents makes the promises while the other merely accepts or rejects them. For tackling that we propose stage 3 in which we allow the agents to respond to promises with complementary promises that extend previous promises. For example, in the dialogue

- $tell(x, y, request(give(r, (13, 15))), d(3), 1)$
- $tell(y, x, promise(r, (10, 12), (13, 15)), d(3), 2)$
- $tell(x, y, promise(r, (15, 17), (10, 12)), d(3), 3)$

agent x responds to y's promise with another promise which practically says: "I will give you the resource r for the interval (10,12) if you give me r for the interval (15,17) and of course for the interval (13,15) as you have already promised".

Hence each new promise extends the previous one and allows to reach agreements in which more than one exchange of a resource is required. For example, the problem that stage 2 could not solve $I_y = \{< a, r, 2, 10, 15 >, < c, r, 2, 15, 20 >\}$, $I_x = \{< b, r, 5, 10, 20 >\}$, $R_y = \{have(r, 13, 17))\}$, $R_x = \{have(r, 10, 13)), have(r, 17, 20))\}$ can be solved by stage 3 through the following dialogue:

- $tell(x, y, request(give(r, (13, 15))), d(3), 1)$
- $tell(y, x, promise(r, (10, 12), (13, 15)), d(3), 2)$
- $tell(x, y, promise(r, (15, 17), (10, 12)), d(3), 3)$
- $tell(y, x, promise(r, (18, 20), (15, 17)), d(3), 4)$
- $tell(x, y, accept(promise(r, (18, 20), (15, 17)), d(3), 5))$

The result of this dialogue is that agent x obtains r for the intervals (13,15) and (15,17) and agent y for the intervals (10,12) and (18,20). After these exchanges both agents can carry out their activities, x in the interval (13,17) and y in the intervals (10,12) and (18,20).

A sequence of promises stops when an agent accepts the last promise or when it cannot make another promise. In the latter case, this agent asks the other agent to change its latest promise. If the other agent cannot make another promise then it asks the first agent to change its previous promise. Thus, the request for change propagates backwards until the initial promise is reached.

4.2 Stage Policy

The agent policy can be represented by 15 move generation rules. The first rule suggests that if an agent receives a request for a resource, has the resource during the requested interval and can give it away (either because he doesn't need it or because he can change his schedule) then he accepts the request. The second rule suggests that if the agent doesn't have at all the requested resource then he refuses the request. The third rule is applicable when the agent has the resource, needs it and there is an exchange that that he could suggest in order to carry out his activity in during another time interval. In such a case he makes a promise to the other agent. Rule 4 is for the exact opposite situation, namely when the agent cannot find a valid deal to suggest and consequently refuses the request.

Rules 5 to 7 are applicable when an agent receives a promise in a dialogue that has been initiated by a request made by itself. In order to evaluate this promise, the agent recalls all the previous promises related to this promise and checks whether the resource exchange that they suggest makes its activity feasible. If that's the case then it accepts the promise otherwise it tries to find a new time interval to use for a new promise. If it cannot find such an interval then it asks from the other agent to change its promise.

Rules 8 to 10 are applicable when an agent receives a promise in a dialogue that has been initiated by a request made by the other agent. In order to evaluate this promise, the agent recalls all the previous promises related to this promise and checks whether the resource exchange that they suggest makes any of its activities not feasible. If that's not the case then it accepts the promise otherwise it tries to find a new time interval to use for a new promise. If it cannot find such an interval then it asks from the other agent to change its promise.

Rules 11 to 12 are applicable when an agent receives a change_promise message in a dialogue that has been initiated by a request made by itself. The agent recalls all the previous promises related to this promise and tries to find a new time interval to use for a new promise. If it cannot find such an interval then it asks from the other agent to change its promise.

Finally, rules 13 to 15 are applicable when an agent receives a change_promise message in a dialogue that has been initiated by a request made by the other agent. The agent recalls all the previous promises related to this promise and tries to find a new time interval to use for a new promise. If it cannot find such an interval then it asks from the other agent to change its promise. If there is no such promise then it refuses the other agent's request.

4.3 Stage Properties

The dialogues that can be generated by stage 3 conform to the protocol of figure 3. As in stage 2 termination is ensured by making the dialogue terminate after a certain time T_{max} and by not allowing repetition of dialogue moves. The range

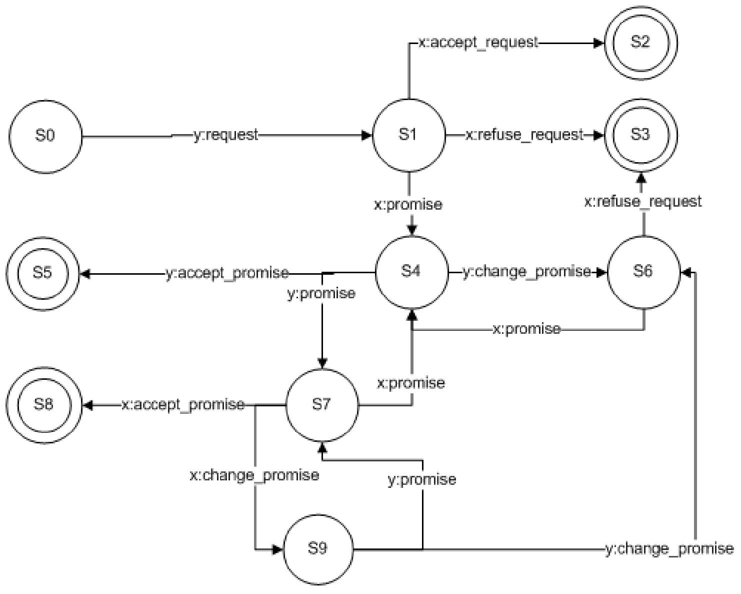


Fig. 1. Stage 3 protocol.

of allocation problems that can be solved by the third stage is defined by the following theorem:

Theorem: If a system consists of two agents x,y then for all system resources r, all activities a assigned to x, all times τ and all intervals $[T_s, T_e]$ such that

$K_{x,\tau} \vdash miss(r,(T_s,T_e),a) \wedge K_{y,\tau} \vdash need(r,(T_s,T_e),b) \wedge ($
$\exists [T_{s1},T_{e1},T_{s2},T_{e2},...,T_{s(2k+1)},T_{e(2k+1)}],\ [a_1,a_2,...,a_k],\ k \in N,\ |$
$need(r,(T_{s2},T_{e2}),a_1) \wedge\ need(r,(T_{s4},T_{e4}),a_2) \wedge,...,\ \wedge need(r,(T_{s2k},T_{e2k}),a_k) \wedge$
$K_{y,T} \cup \{give\_away(r,(T_s,T_e)), obtain(r,(T_{s1},T_{e1})), \{give\_away(r,(T_{s2},T_{e2})),$
$obtain(r,(T_{s3},T_{e3})),...,give\_away(r,(T_{s2k},T_{e2k})), obtain(r,(T_{s(2k+1)},T_{e(2k+1)}))\}$
$\vdash feasible(b,(T_{s'},T_{e'})) \wedge feasible(a_1,(T_{s1'},T_{e1'})) \wedge$
$feasible(a_2,(T_{s2'},T_{e2'})) \wedge,...,\wedge feasible(a_k,(T_{sk'},T_{ek'})) \vee$
$\exists [T_{s1},T_{e1},T_{s2},T_{e2},...,T_{s(2k)},T_{e(2k)}],\ [a_1,a_2,...,a_{k-1}],\ k \in N,\ |$
$need(r,(T_{s2},T_{e2}),a_1) \wedge need(r,(T_{s4},T_{e4}),a_2) \wedge,...,\ \wedge$
$need(r,(T_{s(2k-2)},T_{e(2k-2)}),a_{k-1}) \wedge K_{y,T} \cup$
$\{give\_away(r,(T_s,T_e)), obtain(r,(T_{s1},T_{e1})), \{give\_away(r,(T_{s2},T_{e2})),$
$obtain(r,(T_{s3},T_{e3})),...,give\_away(r,(T_{s2k},T_{e2k}))\}$
$\vdash feasible(b,(T_{s'},T_{e'})) \wedge feasible(a_1,(T_{s1'},T_{e1'})) \wedge$
$feasible(a_2,(T_{s2'},T_{e2'})) \wedge,...,\wedge$
$feasible(a_{k-1},(T_{s(k-1)'},T_{e(k-1)'})))$

for some $(T_{s'}, T_{e'}, T_{s1'}, T_{e1'}, ..., T_{sk'}, T_{ek'})$ or $(T_{s'}, T_{e'}, T_{s1'}, T_{e1'}, ..., T_{s(k-1)'}, T_{e(k-1)'})$, there exists a dialogue d induced by the policy of stage 3, starting at time τ and ending at time τ' such that $K_{x,\tau'} \vdash need(r, (T_s, T_e), a, \tau') \wedge K_{y,\tau'} \vdash indiff(r, (T_s, T_e))$.

The intuitive meaning of this theorem is that if an agent x misses a resource r for an activity a and another agent y needs r for some activity b and there is a set of time intervals whose sequential exchange between x and y makes feasible x's activity a, y's activity b and all y's activities whose feasibility might be disturbed by some step of the exchange sequence, then there exists a dialogue d induced by the policy of stage 3 that generates this exchange sequence and results in x acquiring the missing resource with y not needing it anymore.

5 Conclusions

In this paper we presented an approach in solving the Temporal Resource Reallocation Problem (TRR-P) in multi-agent systems through multi-stage negotiation. Our work extended the negotiation framework defined in [1] by proposing a third negotiation stage that overcomes the limitations of the two existing ones and solves a larger variety of TRR-P instances. The key enabler of this advanced effectiveness was our allowing the agents to exchange sequences of complementary promises and to achieve thus more complex resource exchange deals.

Our third stage comprised a protocol to which the dialogues generated by the stage conform and a policy the agents should follow in order to implement this protocol.The semantic representation of the policy was omitted due to space limitations.

Our current work focuses in the development on a variation of the multi-stage negotiation framework in which agents negotiate not only over resources but also over tasks. Another dimension we also examine is negotiation over divisible resources.

References

1. Fariba Sadri, Francesca Toni, Paolo Torroni: Minimally intrusive negotiating agents for resource sharing. IJCAI 2003: 796-804
2. Sadri, F.,Toni, F. & Torroni, A multi-stage negotiation architecture for sharing resources amongst logic-based agents. Department of Electronics, Computer Science, an Systems, University of Bologna, Italy. DEIS Technical Report DEIS-LIA-02-008, Universita di Bologna, LIA Series No 61, November 2002 (21 pages).