Chapter 10

Polynomial Space Asynchronous Dynamic Reordering

What profit hath a man of all his labour which he taketh under the sun?
[One] generation passeth away, and [another] generation cometh...
There is an evil [which] I have seen under the sun,
as an error [which] proceedeth from the ruler:
Folly is set in great dignity, and the rich sit in low place.
Solomon, Ecclesiastes 1:3,4; 10:5,6, KJV

A second important application of signatures is the dynamic reordering of the agents during search. For designing this technique, I have chosen the next two design principles:

- The technique should combine proposals on reordering from different agents.
- The technique should have polynomial space requirements.

The first design principle is motivated by the fact that agents enforce private problems and no central authority knows enough to decide the best order to choose. Proposals from the agents should therefore be somehow combined. The technique I have obtained is determined by the fact that I started from the first of these principles.

10.1 Combining proposals on reordering from different agents

When I wanted to design a technique combining proposals on reordering from different agents, the immediate idea was to reuse the technique employed to allow agents to combine proposals on instantiation of variables from different agents in AAS. That technique is based on signatures.

Remark 10.1 *Signatures built by agents that change their position do not define causal order of messages when the first element of the pair specifies an ID of the sender.*

A set of proposal sources are therefore defined as a set of *offices*. The offices are statically and totally ordered. Offices can:

- reorder working agents,
- redelegate holders of lower priority offices.

Example 10.13 *A simple example is to consider the case where the* office *is defined by the position that agents can take. Given the previous remark, a simplistic image is that I let the agents build*

proposals on the order, using signatures, but signing in the name of the position that they hold, tagged with the priority of the delegation, instead of signing in their own name.

The offices are denoted with R^k, where k is the position of the office in the total order on offices.

10.2 Reordering and Modern Democracies

After several trial-and-error cycles of efforts to explain this reordering technique in presentations, papers, and personal discussion, I arrived to a solution similar to Lamport's Byzantine generals and Paxos metaphors. The observation is that I can define an understandable modern democracy that illustrates much of the reordering technique introduced next.

10.2.1 Ciglean[1] Democracy.

Following the failure of the Eastern Block and the evaporation of the last traces of the socialist administrative structures, a small unreachable isolated village in a crater on top of a hill develops a new and original democracy that we will call *Ciglean's democracy*.

10.2.1.1 Administrative body in Ciglean

After long delays when nobody wanted to take on administrative tasks, the people decided that everybody should have a responsibility. In Ciglean (100 inhabitants), each citizen must take an administrative office. The citizens of the village are old persons that live alone. To clearly define the responsibility of each administrative office, a strict hierarchy was defined on offices, where the *presidency* is the highest office and also cumulates the function of a prime-minister. The next office is for a second-minister, next for the third-minister, and so on. The lowest priority position is for the hundredth-minister. Each minister knowing the skills of the other citizens can distribute the tasks to lower priority ministers and accomplishes himself any task that cannot be done by any inferior.

The decisions and commands go from the president to all ministers and each minister can give commands to lower priority ministers. There exists only one semicircular road in Ciglean and all the houses are numbered distinctly. Initially the administrative positions were allocated according to the number of the house of the citizen. The president was the happy owner of the house numbered 1 (the influent peasant that had the idea of the new administration).

The phenomena observed in the first years of the Ciglean's Democracy was that the last minister had to repair the roads, to paint the trees, to gather the taxes, etc. while all the other ministers used to simply pass the commands that they received to their inferiors and were doing only accidentally some useful administrative work. Since there is too much work to do for a single person, the road of the village kept degrading each year.

10.2.1.2 Legislative body of Ciglean

After two years, the whole population of the village meeting each Sunday evening in the local church accepted that inevitably a rotation of the positions is unavoidable in order to make Ciglean's road practicable. A system of elections was put up. Since it was not known in advance how efficient each citizen is in a given office, a mechanism is defined to allow reordering of offices whenever needed.

As the citizens did not want to change the tasks of administrative offices with which whey were used, a parallel system was designed, where a hierarchical legislative body has the right to decide whenever certain redelegations were felt needed. The person being president at the moment, has proposed for the legislative body a similar structure to the one of the administrative body.

A set of 99 legislative offices is organized. The priorities of these legislative offices are totally and statically ordered. The highest priority legislative authority is the whole population of the

[1] Ciglean is a small village in the Sălaj/Szilagy county.

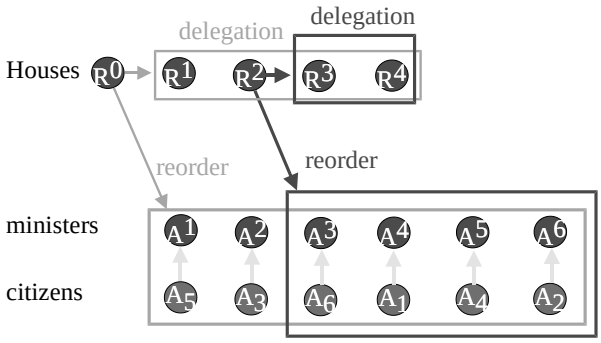


Figure 10.1: Democracy: People choose president and delegate deputies. Deputies approve cabinet and delegate parliamentary committees.

village meeting each Sunday evening in the courtyard of the church. The other legislative offices are called the First-House, the Second-House, etc. The parliament of Ciglean has 98 Houses.

Each time somebody asks, the village votes a new delegation of the administrative body that can change the holder of any administrative office. Each k^{th}-House can reorder the holders of any t^{th}-minister where $t>k$. For example, no House can redelegate the president and the 2^{nd}-House cannot redelegate the second-minister and the president.

With the opportunity of an administrative reordering vote, each legislative office can also redelegate the holders of all lower priority legislative offices by redefining the citizens that have to sign in order to enable the decision of each given House. These notions are also exemplified in Figure 10.1.

10.2.1.3 Signatures

Each decision of a legislative office is signed with a set of stamps that the office receives on its delegation. Each time a certain legislative House is delegated (its members are listed) the new holders of the House receive a copy of all the stamps of the authority delegating them plus a stamp naming the authority taking the decision and the reference number of the decision. The new members of the House receive a new set of forms where they can fill up decisions starting with reference number 1 and incrementing the decision reference number after each new decision.

Each decision for redelegating holders of administrative offices is signed with all the stamps of the House taking the decision, as well as with the name of the House and the reference number of the decision.

Whenever the members of a House or some ministers see a signature that is stronger than the one delegating her to a certain office, the old delegation is forgotten and the corresponding stamps are recycled.

10.2.1.4 Self Redelegation

Citizens of the village sometime leave for a while and they want to redelegate another group to hold the House. When this problem first appeared, the solution that was chosen is to let all the stamps of the House be simply handled to the new holders and the empty decision forms of the old holders are transmitted to the new ones (with the old reference numbers).

There are continuous discussions in Ciglean whether the general assembly of the village should be enabled to similarly redelegate its authority to some single person or to some group during periods of intensive agricultural work, but the citizens are afraid that the power will be retained by some person that will never re-convoke and yield back the main legislative power to the whole population.

10.2.2 Responsiveness

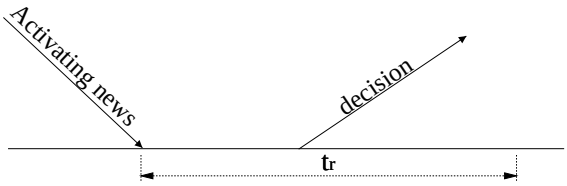


Figure 10.2: To ensure termination, the responsiveness of the agents is limited to a timeout t_r.

To ensure that no malicious House would keep reordering the ministers for ever such that no work could ever be done, a certain limit is put on the validity of a House. A k^{th}-House can only issue new decisions within a time t_r after an information concerning (see Figure 10.2):

- the need of a new administrative work,
- a decision of a higher priority House
- a decision of a t^{th}-minister, with $t \leq k$

10.3 Polynomial Space Asynchronous Reordering

Let us see how Ciglean's democracy applies to our DisCSP framework. There is one major issue to be solved for allowing agents to asynchronously propose new orders on themselves. According to (Collin *et al.* 1991b), in general, infinite loops cannot be avoided if always[2] more than one agent involved in a conflict can change their instantiation. The agents in a conflict must be able to eventually coherently decide which one of them should change its instantiation (an order).

10.3.1 Reordering with restarts

A simple way to dynamically reorder agents in asynchronous search (e.g. for efficiency) is to intermittently synchronize among all agents a consistent view of what the order is. We call it *reordering with restarts*. The search is complete if the delay between two reorderings increases asymptotically (e.g. monotonically) to ∞. In centralized settings this is related to the work in (Baptista & Marques-Silva 2000). I have first described this technique in (Silaghi *et al.* 2001h).

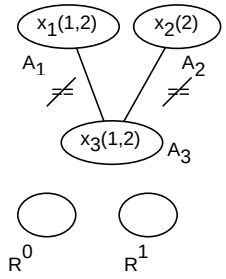
10.3.2 Finite number of reorderings

On the other hand, if we need that agents propose reordering asynchronously (e.g. as in AWC or as required for security in (Silaghi *et al.* 2001c)), the order typically changes before an agreement could be reached. What one can do in this case is to ensure that only *a finite number of orderings* can be considered. One then must enable agents to coherently decide *which of two received orderings is the strongest*.

However, we then put an artificial bound on the number of reorderings proposed by each agent. This would not match the global complexity of a problem but corresponds to a two stage algorithm: an *incomplete min-conflict*-like search for a bounded time, followed if necessary by a version of ABT (e.g. upon AWC, this can be achieved by putting an upper-bound on $k(A_i)$, something like (Yokoo 1993b), and running the ABT which stores only valid nogoods and tags assignments with counters).

However, the initial incomplete search can be performed equally with Distributed Breakout or Distributed Stochastic Search.

[2]There is not a problem if the event eventually occurs but it should not always occur.



$A_1/\mathbf{A^1}$ —**ok?**$\langle x_1,1,1\rangle(A_1,A_2,A_3)$→ A_3
$A_3/\mathbf{A^3}$ —**heuristic**$\langle x_1,1,1\rangle\langle x_3,2,1\rangle(A_1,A_2,A_3)$→ A_3
$A_2/\mathbf{A^2}$ —**ok?**$\langle x_2,2,1\rangle(A_1,A_2,A_3)$→ A_3
$A_3/\mathbf{A^3}$ —**nogood**$\neg(\langle x_1,1,1\rangle,\langle x_2,2,1\rangle)(A_1,A_2)$→ A_2
$A_2/\mathbf{A^2}$ —**add-link**→ A_1
$\mathbf{R^1}$ —**reorder** $(A_1,A_3,A_2)|1:1|$→ A_3
$\mathbf{R^0}$ —**reorder** $(A_3,A_1,A_2)|0:1|$→ A_3
$\mathbf{R^1}$ —**reorder** $(A_1,A_3,A_2)|1:1|$→ A_2
$\mathbf{R^0}$ —**reorder** $(A_3,A_1,A_2)|0:1|$→ A_1
$A_3/\mathbf{A^1}$ —**ok?**$\langle x_3,1,2\rangle(A_3,A_1,A_2)|0:1|$→ A_1
$A_3/\mathbf{A^1}$ —**ok?**$\langle x_3,1,2\rangle(A_3,A_1,A_2)|0:1|$→ A_2

Figure 10.3: Simplified example for ABT with random reordering based on dedicated agents. A_i/A^j in the left column shows that A_i had the position j when it has sent the message. The time increases downwards.

10.3.3 Scalable Dynamic Reordering with Total Order

Now we show how the signatures described in the previous section help agents during the search to asynchronously and concurrently propose new orders on themselves. In the next subsection we describe a didactic simple version that needs additional specialized agents.

10.3.4 Reordering with dedicated agents

The signatures in the previous sections are built by proposal sources ordered statically. We only know to use signatures built by statically ordered proposal sources. The idea behind asynchronous reordering in asynchronous search is to consider the proposal sources for ordering as additional abstract agents. Their activity can be delegated to different physical agents along the search. This seems complicated to apprehend at once and we start by first considering that the proposal sources for the ordering are a set of external distinct agents. Besides the agents $A_1, ..., A_n$ in the DisCSP we want to solve, we consider that there exist $n-1$ other agents, $R^0, ..., R^{n-2}$, solely devoted for reordering the agents A_i.

Definition 10.1 *An* **ordering** *is a sequence of distinct agents* $A_{k_0}, ..., A_{k_n}$.

One has to attach to each proposal on ordering a *signature* as defined in Chapter 8. The *proposal sources* for the ordering on agents are the agents R^i, where $R^i \prec R^j$ if $i<j$ and $k_0^{\text{order}}=0$. R^i is the *proposal source* that when its strongest known ordering is o, can propose orderings that reorder only agents on positions p, $p > i$.

An ordering may place an agent A_i on different positions j, e.g. $j \neq i$. Let us assume that the agent A_l, whose strongest known ordering is o, believes that the agent A_i, owning the variable x_i, has the position j. In algorithms played by A_l, we refer A_i with $A_i^j(o)$. For simplifying notations when subscripts, superscripts and/or parameters are unambiguous from context, we refer $A_i^j(o)$ as either A_i, $A^j(o)$, A^j or A_i^j. The variable x_i is also referred to within algorithms played by A_l as either x_i, $x^j(o)$, x^j or x_i^j.

In difference from ABT, all the agents owning variables related by a constraint with the variable of the current agent are inserted in *outgoing links* at initialization. However, as in ABT, **ok?** messages are sent only to those agents in *outgoing links* that have lower priority than the current agent.

Definition 10.2 (Known order) *The known order of* R^i *(respectively* A^i*) is the order* o *with the strongest signature among those proposed by the agents* $R^k, 0 \leq k < i$ *and received by* R^i *(respectively by* A^i*).* A^i *has the position* i *in* o*. This ordering is referred to as the* known order of R^i *(respectively of* A^i*).*

Remark 10.2 *As mentioned in Section 8.2.1, alternatively to using signatures, agents could have used simple counters. An agent would have to store all the last orderings it was proposed and (e.g.)*

take as known order *the one proposed by the lowest priority R^i that is consistent with all the orders proposed by $R^j, j < i$.*

Rule 8 (Proposed order) *An ordering, o,* proposed by *R^i is such that the agents placed on the first i positions in the* known order of *R^i must have the same positions in o. o is referred to as the* proposed order of *R^i.*

Let us consider two different orderings, o_1 and o_2, with their corresponding signatures: $O_1 = \langle o_1, h_1\rangle$, $O_2 = \langle o_2, h_2\rangle$; such that $|h_1| \leq |h_2|$. Let p_1^k=$|a_1^k$:$b_1^k|$ and p_2^k=$|a_2^k$:$b_2^k|$ be the pairs on the position $k, k > 0$ in h_1 respectively in h_2.

Definition 10.3 (Reorder position) *Let u be the lowest position such that p_1^u and p_2^u are different and let $v = |h_1|$. The* **reorder position** *of h_1 and h_2, R(h_1,h_2), is either $min(a_1^u, a_2^u) + 1$ if $u > v$, or $a_2^{v+1} + 1$ otherwise. This is the position of the highest priority agent that may have been reordered between h_1 and h_2.*

New optional messages for reordering are:

- **heuristic** messages for heuristic dependent data, and
- **reorder** messages announcing a proposal ordering, $\langle o, h\rangle$.

An agent R^i announces its proposed order o by sending **reorder** messages to all agents $A^k(o), k > i$, and to all agents $R^k, k > i$. Each agent A_i and each agent R^i has to store its *known order* denoted $O^{crt}(A_i)$ respectively $O^{crt}(R^i)$. For allowing asynchronous reordering, each **ok?** and **nogood** message receives as additional parameter an order and its signature (see Algorithm 11). The **ok?** messages hold the strongest *known order of* the sender. Each **nogood** message sent from A^j to A^i holds the order, in agreement with $O^{crt}(A^j)$, that A^j *believes* to be $O^{crt}(A^i)$. This ordering denoted $O_i^{crt}(A^j)$ consists of the first i agents in the $O^{crt}(A^j)$ and is tagged with a signature obtained from the signature of its O^{crt} by removing all the pairs $|a$:$b|$ where $a \geq i$.[3] For example, in Figure 10.3 the 4th message contains $O_2^{crt}(A^3)$=(A_1, A_2).

When A_i receives a message which contains an order with a signature h that is stronger than the signature h^* of $O^{crt}(A_i)$, let the *reordering position* of h and h^* be I^r. The assignments known by A_i for the variables x^k, $k \geq I^r$, are invalidated.

The agents R^i can modify the ordering in a random manner or according to special strategies appropriate for a given problem.[4] Sometimes it is possible to assume that the agents want to cooperate in order to decide an ordering. The **heuristic** messages are intended to offer data for reordering proposals. These parameters depend on the used reordering heuristic. The **heuristic** messages can be sent by any agent to the agents R^k. **heuristic** messages may only be sent by an agent to R^k within a bounded time, $\mathbf{t_h}$, after having received a new assignment for $x^j, j \leq k$. Agents can only send **heuristic** messages to R^0 within time t_h after the start of the search. Any **reorder** message is sent within a bounded time $\mathbf{t_r}$ after a **heuristic** message is received or from start.

Besides C_k^{order} and $O^{crt}(R^k)$, the other structures that have to be maintained by R^k, as well as the content of **heuristic** messages depend on the reordering heuristic. The space complexity for A^k remains the same as with ABT.

10.3.5 ABT with Asynchronous Reordering (ABTR)

In fact, we have introduced the physical agents R^i in the previous subsection only in order to simplify the description of the algorithm. The agents R^i can be seen as roles or offices. Any of the agents A_i or other entity can be delegated to act for any R^j. When proposing a new order, R^i can also simultaneously delegate the identity of $R^i, ..., R^{n-2}$ to other entities, P_k, by attaching a sequence $R^0 \rightarrow P_{k_0}, ..., R^{n-2} \rightarrow P_{k_{n-2}}$ to the ordering.

[3] The agents absent from the ordering in a nogood are typically not needed by A^i. A^i receives them when it receives the corresponding **reorder** message.

[4] e.g. first the agents forming a coalition with R^i.

```
when received (ok?,⟨x_j, d_j, c_{x_j}⟩*,⟨o, h⟩*) do
    *getOrder(⟨o, h⟩);
    if(old c_{x_j}) return;
    add(x_j,d_j,c_{x_j}) to agent_view;
    reconsider stored and invalidated nogoods;
    check_agent_view;
when received (nogood,A_j,¬N*,⟨o, h⟩*) do
    *getOrder(⟨o, h⟩);
    if I am not CEA(¬N) then return;
    discard ← false;
    if ((((⟨x_i, d, c⟩∈N ∧ (A_i knows (M→(x_i≠d))) ∧ ¬(better ¬N than ¬M))
            ∨ invalid(¬N))) then
        if (I do want to discard ¬N) then
            discard ← true;
        else
            store ¬N as redundant constraint;
    else
        put ¬N in nogood-list for x_i=d;
    if ¬ discard then
        for every ⟨x_k, d_k, t_k⟩ ∈ N, where x_k is not connected do
            send add-link to A_k;
            add ⟨x_k, d_k, t_k⟩ to agent_view;
        add the other new assignments in ¬N to agent_view;
25.1    reconsider stored and invalidated nogoods;
    old_value ← current_value; check_agent_view;
    when old_value = current_value *and if A_j has lower priority than A_i*
25.2    send (ok?,⟨x_i, current_value, C^i_{x_i}⟩*,O^{crt}*) to A_j;
```

Algorithm 25: Procedures of A_i for receiving messages in ABTR. Code between '*' is added to ABT.

Example 10.14 *An example of order is:*

$$A_3, A_1, (A_4, A_2) : R^0 \rightarrow \{A_1, A_3\}, R^1 \rightarrow \{A_3, A_4\}, R^2 \rightarrow \{A_4\} : |0:3|1:15|$$

- $A_3, A_1, (A_4, A_2)$ *is a sequence defining a total order on the agents. It is composed of*
  - *a guideline:* A_3, A_1 *should be the first two agents, and*
  - *a proposal:* A_3, A_1, A_4, A_2.
- $R^0 \rightarrow \{A_1, A_3\}, R^1 \rightarrow \{A_3, A_4\}, R^2 \rightarrow \{A_4\}$ *is a delegation of proposal sources for reordering.*
- $|0:3|1:15|$ *is a signature defining the priority of the order.*

At a certain moment, due to message delays, there can be several entities believing that they are delegated to act for R^i based on the ordering they know. However, any other agent can coherently discriminate among messages from simultaneous R^is using the signatures that R^is generate. The R^i themselves coherently agree when the corresponding orders are received. The delegation of $R^j, j{>}i$ from a physical entity to another poses no problem of information transfer since the counter C_j^{order} of R^j is reset on this event. The counter C_i^{order} of a new R^i delegated by a previous different R^i is set to k where $|i{:}k|$ is a pair in the signature that tags the new received ordering.

For example, in Figure 10.4 we describe the case where the activity of R^i is always performed by the agent believing itself to be A^i. R^i can send a **reorder** message within time t_r after an

procedure check_agent_view do
  when agent_view *and* current_value *are not consistent*
    if *no value in* D_i *is consistent with* agent_view **then**
      backtrack;
    else
      select $d \in D_i$ where *agent_view* and d are consistent;
      current_value $\leftarrow$ d; $C^i_{x_i}$++ ;
      send (**ok?**,$\langle x_i, d, C^i_{x_i}\rangle$*,$O^{crt}$*) to lower priority agents in *outgoing links*;

procedure backtrack do
  nogoods $\leftarrow \{V \mid V =$ inconsistent subset of *agent view*$\}$;
  when *an empty set is an element of* nogoods
    broadcast to other agents that there is no solution;
    terminate this algorithm;
  for every $V \in$ nogoods **do**
    select (x_j,d_j,t_{x_j}) where x_j has the lowest priority in V;
26.1 send (**nogood**,A_i,V*,O^{crt}_j*) to A_j;
    remove (x_j,d_j*,t_{x_j}*) from *agent_view*;
    reconsider stored and invalidated explicit nogoods;
  check_agent_view;
function getOrder$(\langle o, h\rangle) \rightarrow$ *bool //ABTR*
  when h *is invalidated by the signature* O^{crt} **then** return false;
  when *not newer* h *than* O^{crt}
    return true;
  $I \leftarrow$ *reorder position* for h and the signature of O^{crt};
  invalidate assignments for $x^j, j \geq I$;
  $\langle o, h\rangle \rightarrow O^{crt}$;
26.2 make sure that `send` (**ok?**,$\langle x_i,$ some value, $c_{x_i}\rangle$,O^{crt}) will be performed
    to all lower priority agents in *outgoing links*;
  return true;

Algorithm 26: Procedures of A_i in ABTR.

assignment is made by A^i since a **heuristic** message is implicitly transmitted from A^i to R^i. We also consider that A_2 is delegated to act as R^0. R^0 and R^1 propose one random ordering each, asynchronously. The receivers discriminate based on signatures that the order from R^0 is the strongest. The known assignments and nogood are discarded. In the end, the *known order* for A_3 is $(A_3, A_1, A_2)|0:1|$.

Definition 10.4 (Quiescence in ABTR) *By* **quiescence** *of a group of agents we mean that none of them will receive or generate any valid nogoods, new valid assignments,* **reorder** *messages or* **add-link** *messages.*

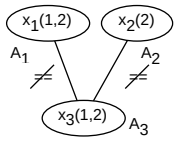
Property 10.1 $\forall i$*, in finite time* t^i *either a solution or failure is detected, or all the agents* $A^j, 0<j\leq i$ *reach quiescence in a state where they are not refused an assignment satisfying the constraints that they enforce and their* agent view.

Theorem 10.2 *ABTR is correct, complete and terminates.*

Proof. **Completeness:** All the nogoods are generated by logical inference from existing constraints. Therefore, if a solution exists, no empty nogood can be generated.

No infinite loop: This is a consequence of the Property 10.1 for $i = n$.

Correctness: All assignments are sent to all interested agents and stored there. At quiescence all the agents know the valid interesting assignments of all predecessors. If quiescence is reached without detecting



$A_1/\mathbf{A^1}/R^1$ —**ok?**$\langle x_1,1,1\rangle(A_1,A_2,A_3)$→ A_3
$A_2/\mathbf{A^2}/R^0/R^2$ —**ok?**$\langle x_2,2,1\rangle(A_1,A_2,A_3)$→ A_3
$A_3/\mathbf{A^3}/R^3$ —**nogood**$\neg(\langle x_1,1,1\rangle,\langle x_2,2,1\rangle)(A_1,A_2)$→ A_2
$A_1/\mathbf{R^1}/A^1$ — **reorder** $(A_1,A_3,A_2)|1:1|$→ A_3
$A_2/\mathbf{R^0}/A^3/R^2$ — **reorder** $(A_3,A_1,A_2)|0:1|$→ A_3
$A_1/\mathbf{R^1}/A^1$ — **reorder** $(A_1,A_3,A_2)|1:1|$→ A_2
$A_2/\mathbf{R^0}/A^3/R^2$ — **reorder** $(A_3,A_1,A_2)|0:1|$→ A_1
$A_3/\mathbf{A^1}/R^1$ —**ok?**$\langle x_3,1,2\rangle(A_3,A_1,A_2)|0:1|$→ A_1
$A_3/\mathbf{A^1}/R^1$ —**ok?**$\langle x_3,1,2\rangle(A_3,A_1,A_2)|0:1|$→ A_2

Figure 10.4: Example for ABTR with random reordering. R^i delegations are done implicitly by adopting the convention "A^i is delegated to act for R^i". Left column: $A_i/A^j/R^{i_1}/R^{i_2}$... shows the roles played by A_i when the message is sent. In bold is shown the capacity in which the agent A_i sends the message. The addlink is not shown.

function getOrder$(\langle o,h\rangle) \rightarrow$ *bool*
  when *h is invalidated by the signature of* O^{crt} **then** return false;
  when *not newer h than* O^{crt} **then** return true;
  $O^{crt} \leftarrow \langle o,h\rangle$;
  invalidate assignments for $x^j(o), j > v$ (remove from *agent view*);
27.1 make sure that `send` (**ok?**,$\langle x_i$, some value, $c_{x_i}\rangle$,O^{crt}) will be performed
    to all lower priority agents in *outgoing links*;
  when *I am* R^w **then** $C_w^{\text{order}} \leftarrow (\exists|w{:}k|{\in}h)?k:0$;
  return true;

Algorithm 27: *Procedure of* A_i^v *for receiving new orderings in ABTR1.*

an empty nogood, then according to the Property 10.1, all the agents agree with their predecessors and the set of their assignments is a solution. □

As we show in chapter 17 (Silaghi *et al.* 2001c), ABTR is required for improving security in algorithms for automated negotiations. In the sequel of this chapter we deal with efficiency issues.

10.3.6 Saving effort across reordering

We let agents maintain their current assignment when a new order is received. If the old order known by A_i was $\langle o',h'\rangle$, after receiving a new order, $\langle o,h\rangle$, $A_i^j(o)$ removes from its *agent view* only those assignments $\langle x^k(o),v,c\rangle$ where $k > j$. Therefore, $A_i^j(o)$ discards only nogoods containing some assignment $\langle x^k(o),v,c\rangle, k > j$, since the validity of such nogoods cannot be checked in o. A_i (re)sends its current assignment via **ok?** messages to any agent $A^u(o)$ in its *outgoing list* where $u > j$. This version of ABTR is referred to as ABTR1.

Theorem 10.3 *ABTR with assignments reuse across reordering (ABTR1) has polynomial space, is correct, complete and terminates.*

Proof. Since the assignments are resent, the only difference with ABTR is that some additional nogoods are stored. These additional nogoods are consistent with the Theorem 10.2. The space remains polynomial since only one assignment is valid and only one nogood continues to be stored for each value of each variable. The space complexity is not modified. □

10.4 Partial Ordering in Asynchronous Search

(Ginsberg & McAllester 1994; Bliek 1998a) show generalizations of centralized backtracking where the order on agents does not need to be total. Whenever ties have to be broken, an order is set on the conflicting variables, and the order can be removed when the information on that conflict is discarded.

A version of the GPB algorithm proposed in (Bliek 1998a) is given in Section A.3.2. GPB in its most general version is not systematic, but a systematic version called GPB' is the one that we will actually *asynchronize*. The natural problem that anybody will pose himself is whether an equivalent asynchronous technique could be developed. The solution proposed is called Partial Order Asynchronous Backtracking (POAB).

In POAB, each agent works asynchronously and concurrently. The agents propose instantiations for the variables that they can modify and announce these proposals to all agents interested in these variables.

Each agent A_i stores a partial order S_i on all the agents. A_i reasons on the aggregates stored in his view and that are proposed from any agent A_z such that $A_i \not<_S A_z$.

Remark 10.3 *Some ordering conditions may have to be initialized and maintained in order to ensure that each variable x can be modified by an agent placed before or identically to any agents that must enforce constraints on x.*

Any nogood γ discovered by A_i is sent to all agents with proposals found in γ. These agents start a *coMechanism* (see Section 8.4) and decide a conclusion A_j for γ. Each agent A_k involved in γ adds to S_k the condition $A_k <_S A_j$, and these conditions are broadcasted to all other agents.

Agents receiving nogoods with a new variable x send **add-link** messages to all agents that are modifiers of x. Whenever an aggregate is invalidated, the agents receiving it discard the ordering conditions induced by invalidated nogoods.

Each pair of agents maintains a counter for their joint decision on enabling or disabling an order on the corresponding pair. The value of the counter tags their decision. Each time a pair of agents synchronize a decision on an order with a *coMechanism*, they also synchronize their partial orders, S. When the decision is taken, it is broadcasted together with the composed partial order.

Each agent stores a matrix containing the last ordering decision for each pair of agents, as revealed by the tag of the decision.

Remark 10.4 *Due to the fact that agents synchronize their partial order when they decide an order between them, it can be shown that there is no way for a circularity to appear (the rules of GPB' are respected).*

The order condition broadcasted, $A_k < A_j$, are usable for detecting ordering conditions that have to be discarded or added:

- any agent A_z such that $A_z > A_j$, discards all the ordering in which he is not constrained by valid nogoods.
- any agent A_z such that $A_z > A_j$, adds $A_z > A_x$ for all $A_x < A_j$.

The pseudo-code of the described technique is given in Algorithm 28.

10.5 Summary

An important application of the signatures is the dynamic reordering of the agents. I have noted that the order on agents can be treated as a shared resource controlled by a hierarchy of authorities. This chapter describes and explains how reordering is performed.

```
when received (ok?,⟨x_j, s_j, c_{x_j}⟩) do
    if valid then
        insert aggregate in view; remove invalidated nogoods and conditions;
    if agent generating received aggregate is not positioned with less priority then
        check_agent_view;
when received (nogood,A_j,¬N) do
    update view and request addition of needed links;
    if nogood is valid (this implies one nogood per proposal) then
        store it;
    else
        check_agent_view;
        return;
    synchronize with the other agents generating aggregates in ¬N do
        exchange view and ordering conditions;
        decide a conclusion for ¬N;
        increase counters of the local conditions;
    broadcast obtained ordering conditions;
    if I generated the conclusion then
        check_agent_view;
when received (ordering conditions) do
    foreach received A_j > A_k do
        if A_i > A_j then
            remove all conditions;
            add conditions in stored nogoods;
            add condition A_i > A_k;
    store all valid received conditions;
procedure check_agent_view do
    if can find an acceptable proposal satisfying any non-successor then
        broadcast it;
    else
        backtrack;
procedure backtrack do
    find nogoods and send each of them to all agents involved in it;
```

Algorithm 28: Procedures of A_i for receiving messages in POAS. Proposals are tagged with counters. The technique mentioned as alternative to signatures is used for combining proposals (see Section 8.2.1.)