Chapter 6

Nogood elimination in asynchronous search – Polynomial space requirements

"Eighty percent of success is just showing up"
Woody Allen

HERE we will see how one can remove nogoods from asynchronous search. I will describe two alternatives that I have proposed so far for ensuring polynomial space requirements in asynchronous complete search.

- tagging of assignments (Silaghi *et al.* 2000a). This technique is efficient and is the one that I use to recommend.
- retransmission of messages (Silaghi *et al.* 2000a). It is recommended in AAS0 for cases where one wants to discard valid nogoods. It is more expensive due to required communication. One can nevertheless design algorithms where retransmission is used extensively (for the elimination of any nogood).

6.1 Nogood elimination in ABT

In this part we adapt successful techniques from centralized CSP to asynchronous search. The version of ABT presented in Section 4.2 requires a space whose theoretical complexity in each agent is exponential in the global problem. Therefore not only the computation time but also the required hardware resources cannot scale well with the size of the problems.

It was quite early (Havens 1997; Hamadi & Bessière 1998; Hamadi 1999b; Yokoo *et al.* 1998) noticed that ABT could be modified into a polynomial space asynchronous algorithm. (Silaghi *et al.* 2000a) that introduces the Asynchronous Aggregation Search (AAS) algorithm presented later, and which is an extension of ABT with nogood elimination, showed how deadlocks can be avoided in such algorithms by time-stamping assignments. This first solution was also transmitted by Makoto Yokoo to the authors of (Hamadi & Bessière 1998), as mentioned in (Bessière *et al.* 2001). Time-stamping is a mechanism that defines an ordering.

6.1.1 Ordering assignments by tags

We assume neither FIFO nor Causal Order delivery channels. In our version a **local counter**, $C_i^{x_i}$, in each agent is incremented each time a new instantiation is chosen, and its current value **tags** each generated assignment.

when received *(***ok?**,$\langle x_j, d_j*, c_{x_j}*\rangle$*)* **do**
  __if__(old c_{x_j}) return;
  add $\langle x_j, d_j*, c_{x_j}*\rangle$ to *agent_view*;
  reconsider stored and invalidated nogoods; **check_agent_view**;

when received *(***nogood**,A_j,$\neg N$*)* **do**
  *__if__ ((($\langle x_i, d, c\rangle \in N \wedge$ (A_i knows ($M \rightarrow (x_i \neq d)$))) $\wedge \neg$(better $\neg N$ than $\neg M$))
    $\vee$ invalid($\neg N$))) **then**
    if *(I do not want to discard* $\neg N$*)* **then**
      when $\langle x_k, d_k*, t_k*\rangle$, *where* x_k *is not connected, is contained in* $\neg N$
        send **add-link** to A_k; add $\langle x_k, d_k*, t_k*\rangle$ to *agent_view*;
      store $\neg N$*;
    else
      when $\langle x_k, d_k*, t_k*\rangle$, *where* x_k *is not connected, is contained in* $\neg N$
        send **add-link** to A_k; add $\langle x_k, d_k*, t_k*\rangle$ to *agent_view*;
      put $\neg N$ in *nogood-list* *for x_i=d*;
      *add the other new assignments in $\neg N$ to *agent_view**;
17.1 *reconsider stored and invalidated nogoods*;
  old_value $\leftarrow$ *current_value*; **check_agent_view**;
  when old_value = current_value
17.2   send (**ok?**,$\langle x_i$, *current_value**, $C^i_{x_i}*\rangle$) to A_j;

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?**,$\langle x_i, d*, C^i_{x_i}*\rangle$) 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 $\langle x_j, d_j*, t_{x_j}*\rangle$ where x_j has the lowest priority in V;
17.3 send (**nogood**,A_i,V) to A_j; remove $\langle x_j, d_j*, t_{x_j}*\rangle$ from *agent_view*;
    reconsider stored and invalidated explicit nogoods;
  check_agent_view;

Algorithm 17: Procedures of A_i for receiving messages in polynomial space ABT (ABTp). The modifications are shown between '*'.

Definition 6.1 (Assignment) *An* assignment *for a variable x_i is a tuple $\langle x_i, v, t\rangle$ where v is a value from the domain of x_i and t is the* tag *value.*

Among two assignments for the same variable, the one with the highest *tag* (attached value of the counter) is the **newest**. Therefore each agent, A_i, knows a local CSP, $\mathbf{CSP(A_i)}$, with variables $\mathbf{vars(A_i)}$. The set of constraints enforced by A_i are denoted $\mathbf{ECSP(A_i)}$ and the set of variables that are involved in ECSP(A_i) is denoted $\mathbf{evars(A_i)}$, where $x_i \in$evars(A_i).

Definition 6.2 (Nogood) *A* nogood *has the form $\neg N$ where N is a set of assignments for dis-*

tinct variables.

The agents answer to received messages according to the Algorithm 17.

Definition 6.3 (Valid assignment) *An assignment $\langle x, v_1, t_1\rangle$ known by an agent A_l is valid for A_l as long as no assignment $\langle x, v_2, t_2\rangle$, $t_2 > t_1$, is received.*

Definition 6.4 *A* **nogood is valid** *if it contains only valid assignments.*

The obtained space complexity remains the one mentioned in (Yokoo 2001).

Property 6.1 *If only one valid nogood is stored for a value then ABT has polynomial space complexity in each agent, $O(dv)$, while maintaining its completeness and termination properties. d is the domain size and v is the number of variables.*

Note that this version of ABT no longer requires FIFO order delivery on communication channels.

6.1.2 Proposal retransmission

A second solution to the nogood elimination deadlock in ABT is to always resend proposals after each change of the view. With this new additional convention, the FIFO delivery of the channels already assumed in unbounded space ABT assures that any value removed by a prematurely discarded valid nogood will be reexplored and therefore rediscovered as infeasible by the search. A nogood for this values is inferred by lower priority agents and received again.

The corresponding procedures are given in Algorithm 18 (ABTr). Its main differences compared to Algorithm 17 consist in not tagging assignments but resending assignments on any modification of a view.[1] No modification of assignments is learned from nogoods.

A nogood received by A_i is invalid in Algorithm 18 when it contains an assignment $\langle x_k, v_k\rangle$, and the *agent view* of A_i contains $\langle x_k, v'_k\rangle$, $v_k \neq v'_k$.

Theorem 6.2 *ABTr is complete, correct and terminates. It has polynomial space requirements.*

Proof. Correctness: Due to the FIFO order on communication channels, all agents receive and hold at quiescence the last assignments involved in their nogoods. Due to the retransmission of assignments after receiving nogoods, all CEAs are informed at quiescence about the assignments involved in their constraints. Whenever an agent is quiescent, it is satisfied by his *agent view* and proposal. Therefore at quiescence all the agents are satisfied by a coherent valuation.
Completeness: All the nogoods are obtained by inference. An empty nogood can only be obtained when no solution exists.
Termination: The termination is clear when recursively one notices that any proposal of A_1 is refused only once, then any proposal of A_2 is refused only onced for a proposal of A_1, etc. The result follows by induction on n.
Polynomial space: Each agent only needs to store one nogood per value. □

One can conjuncture that ABTr needs more messages (due to retransmissions) than what ABTp needs. Instead, ABTp's nogoods are slightly larger due to the additional tags. Similar techniques to ABTr are used in AAS0 (Silaghi 2000b) after valid nogood are removed (Silaghi *et al.* 2000a).

6.2 One shot links

In (Hamadi & Bessière 1998), is described a first attempt to develop algorithms that do not need **add-link** messages. A nice algorithm offering completeness is described in (Bessière *et al.* 2001). The lack of **add-link** messages can bring an improvement for dense graphs (Bessière *et al.* 2001) in simulation systems. Moreover, in real settings and especially in algorithms with nogood elimination, **add-link** messages can generate really useless links on which redundant **ok?** messages

[1] If assignment were not be removed on backtracking, the resending is not compulsory when an assignment is added but no existing assignment is removed, and the existing proposal remains valid.

when received *(***ok?**,$\langle x_j, d_j\rangle$*)* **do**
  add $\langle x_j, d_j\rangle$ to *agent view*;
  reconsider stored and invalidated nogoods; **check_agent_view**;
  *send (**ok?**,$\langle x_i$, *current_value*$\rangle$) to lower priority agents in *outgoing links*//added*;

when received *(***nogood**,A_j,$\neg N$*)* **do**
  ***if** *((($\langle x_i, d, c\rangle \in N \wedge$ (A_i knows ($M \rightarrow (x_i \neq d)$))) $\wedge \neg$(better $\neg N$ than $\neg M$))*
      $\vee$ invalid($\neg N$))) **then**
    if *(I do not want to discard $\neg N$)* **then**
      when $\langle x_k, d_k\rangle$, *where x_k is not connected, is contained in $\neg N$*
        send **add-link** to A_k; add $\langle x_k, d_k\rangle$ to *agent_view*;
      store $\neg N$*;
    else
      when $\langle x_k, d_k\rangle$, *where x_k is not connected, is contained in $\neg N$*
        send **add-link** to A_k;
        add $\langle x_k, d_k\rangle$ to *agent view*;
      put $\neg N$ in *nogood-list* *for x_i=d*;
  old_value $\leftarrow$ *current_value*;
  check_agent_view;
  if old_value = current_value **then**
18.1 send (**ok?**,$\langle x_i$, *current_value*$\rangle$) to A_j*;
  else
    send (**ok?**,$\langle x_i$, *current_value*$\rangle$) to lower priority agents in *outgoing links*;

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;
      // remove: send (**ok?**,$\langle x_i, d\rangle$) 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 $\langle x_j, d_j\rangle$ where x_j has the lowest priority in V;
18.2 send (**nogood**,A_i,V) to A_j;
    remove $\langle x_j, d_j\rangle$ from *agent view*;
  check_agent_view;

Algorithm 18: Procedures of A_i for receiving messages in polynomial space ABT with retransmission (ABTr). Modifications are shown between '*'. Removed lines are preceded by *// remove:*.

flood the network. Indeed, the target agent may have removed conflicting nogoods and he is then no longer interested in some variables. (Maestre 2001) has criticized the **add-link** of the ABT-based algorithms and this has lead us to analyze the possible fixes for ABT, as discussed in the following.

In ABT, the links added dynamically during the search become redundant when nogoods are

discarded. It is known that the nogoods are invalidated on the first change of the tag[2] of an assignment.

Rule 2 (one shot links) *A convention can be established such that an agent eliminates another agent from its dynamically added outgoing-links immediately after an aggregate with a new tag is broadcasted for the variable defining that link.*

Remark 6.1 *If the* one shot links *convention is adopted, the agents receiving a new assignment for a dynamically added link have to forget that link from its incoming links.*

If such a new assignment is received in a nogood, a corresponding **add-link** *is generated immediately after the link is forgotten.*

However, this strategy is acceptable only for problems where the link does not reappear frequently. This technique called *one shot links* has been first described in (Silaghi *et al.* 2001i) for generic extensions of ABT. The efficiency of *one shot links* was first evaluated in (Bessière *et al.* 2002) by comparison with *ABTp* and DDB. (Bessière *et al.* 2002) rename DDB as ABT_2, name *ABTp* as ABT_0 and introduce *one shot links* in ABTp, denoting the obtained technique ABT_1. According to the described tests, ABT_1 has for the generated random problems an efficiency that is situated between ABT_0 and ABT_2.

6.3 Summary

In this chapter we have seen how nogoods can be removed in asynchronous search without losing correctness. The removal of nogoods is essential in real settings where agents have finite resources.

Two alternatives have been highlighted:

- tagging of assignments (Silaghi *et al.* 2000a). This technique is efficient and is the one that I use to recommend.
- retransmission of messages (Silaghi *et al.* 2000a). Recommended in AAS0 for cases where one wants to discard valid nogoods. It is more expensive due to required communication.

[2]Signature in case of aggregates.