Asynchronous PFC-MRDAC±Adopt —Consistency-Maintenance in Adopt—

Marius-Călin Silaghi
Florida Institute of Technology
msilaghi@cs.fit.edu

Abstract

Each agent has private problems. Private concerns can often be formulated in a general framework such as constraint satisfaction (where everything is modeled by either variables, values, or constraints). Often agents need to find agreement with others for the allocation of final resources. The constraint satisfaction (CSPs) is only a special case of optimization. Here it is shown how a very general technique for Distributed CSPs, Replica-based Multiply Asynchronous Search (R-MAS) (comprising ABT,ABTR,AAS,DMAC,DMAC-ABT), can be extended and applied to optimization problems in distributed Weighted CSPs (WCSPs). Centralized WCSPs can be seen as MAX-CSPs where several constraints can link the same variables. PFC-MRDAC is a good approach to MAX-CSPs. How to asynchronize and adapt it to distributed WCSPs? This article describes how asynchronous consistency maintenance can be introduced in Adopt and in Asynchronous Branch&Bound[1].

The main new ideas proposed in this article are that: (a) a concept called Weighted Consistency Nogood (WCN) allows to maintain consistency in DisWCSPs, (b) leading to an asynchronous equivalent of PFC-MRDAC. (c) The feedback that Adopt needs about low bounds can be extracted from such WCNs.

1. Introduction

Everybody has her problems. Private concerns can often be formulated in a general framework such as constraint satisfaction problems (where everything is modeled by either variables, values, or constraints) and then can be solved with any of the applicable CSP techniques. But often one has to find agreements with the other agents for a solution from the set of possible valuations that satisfy her subproblem. The general framework modeling this kind of combinatorial problems is called Distributed Constraint Satisfaction.

In practice we should most often expect to meet an optimization problem rather than a satisfaction problem. Nevertheless the techniques developed for satisfaction problems have proved to be very useful when adapted to fit optimization problems (e.g. PFC-MRDAC [21]).

Distributed Weighted CSPs (DisWCSPs) is a general formalism that can model many negotiation problems and can quantify their privacy requirements. Here it is shown how a very general technique for Distributed CSPs, Replica-based Multiply Asynchronous Search (R-MAS), can be extended and applied to optimization in DisWCSPs.

I show in detail how the technique of consistency maintenance in asynchronous search (DMAC-ABT) can be extended and hybridized with Adopt and Branch&Bound. DMAC-ABT is only a small part of R-MAS, but it is the single one needing modifications. All other techniques of R-MAS (reordering of ABTR, aggregation of AAS) apply almost unchanged to the extension. Namely small details change only in the local computation of AAS aggregates and these details are also discussed here.

2. Distributed Weighted CSPs

Constraint Satisfaction Problems (CSPs) do not model optimization requirements. An extension allowing for modeling some optimization functions is given by Weighted CSPs. When D is an ordered set, $D=\{D_1, D_2, ...D_m\}$, we will denote by $\vec{D}$ the set $D_1 \times D_2 \times \ldots \times D_m$. When $\vec{x}$ is a tuple of assignments to a tuple of variables $\vec{X}$, and $X' \subseteq X$, then $x|_{\vec{X'}}$ is the tuple obtained by projecting $\vec{x}$ unto $\vec{X'}$, namely the tuple of assignments from $\vec{x}$ for the variables in $\vec{X'}$.

Definition 1 (WCSP) *A Weighted CSP is defined by a set of variables $X = \{x_1, x_2, ..., x_m\}$ taking values from a corresponding set of domains $D=\{D_1, D_2, ...D_m\}$ (x_i can take values from D_i) and a set of functions,*

[1] In [35] I present these techniques grouped under the DVR-MAS family of algorithms

$f_1, f_2, ..., f_i, ..., f_n$, *of type* $f_i : \vec{D} \rightarrow \mathbb{R}$ *(not all variables are necessarily used in each function).*

The WCSP consists in finding $\underset{\vec{x} \in \vec{D}}{\operatorname{argmin}} \sum_{i=1}^{n} f_i(\vec{x})$.

A CSP is a particular type of WCSP where the functions f are predicates (aka *constraints*), namely functions with results in the set $\{0, \infty\}$, meaning true respectively false. A solution is then defined as any $\vec{x}$ such that $\sum_{i=1}^{n} f_i(\vec{x}) = 0$.

Recent approaches simplify the CSP-based modeling of problems by extending the traditional prepositional logic formulation of the predicates representing the constraints [5]. In these approaches one can use first order logic predicates as constraints. We consider this to be a useful extension and therefore in each predicate we associate each variable with a *quantifier*: either $\exists$, or $\forall$.

It is known that any maximization problem can be straightforwardly translated into a minimization problem. Also note that, given finite problems with n-ary functions f_i, the functions can be represented by n-dimension matrices with values. To model CSPs with WCSPs, infeasible tuples can be set to ∞, and feasible ones to 0. Weighted CSP can be also distributed.

Definition 2 (DisWCSP) *A Distributed Weighted CSP (DisWCSP) is defined by a set of agents* $A_1, A_2, ..., A_n$*, a set of variables* $X = x_1, x_2, ..., x_m$ *taking values from a corresponding set of domains* $D=\{D_1, D_2, ...D_m\}$*, and a set of functions* $f_1, f_2, ...f_i, ..., f_n$, $f_i : \vec{D^i} \rightarrow \mathbb{R}$, D^i *is the set of domains of the set of variables* $X^i \subseteq X$. A_i *is the only agent that knows* f_i.

The problem is to find $\underset{\vec{x} \in \vec{D}}{\operatorname{argmin}} \sum_{i=1}^{n} f_i(\vec{x}|_{\vec{X^i}})$ *where constraints for the domain of each existentially quantified variable* x_i *can be proposed by at least one agent.*

3. Replica-Based MAS

First we give a definition of a distributed CSP (DisCSP) that unifies most approaches found in the literature.

Definition 3 (DisCSP) *A* **DisCSP** *(A,X,D,M,C) is defined by a set of agents* $A = \{A_1, ..., A_n\}$ *where each agent* A_i *wants to enforce some private constraint* $C_i, C_i \in C$.

The set of shared variables involved in C_i *is* X_i, $X_i \in X$. *The agents negotiate the instantiation of the variables in* X_i *with values from the corresponding domains* D_i, $D_i \in D$, *by either revealing conflicts or by proposing instantiations for a set* M_i *of variables,* $M_i \in M$, $M \subseteq X$.

Any variable x_k *that is involved with an existential quantifier in at least one constraint, has to be in the* M_i *set of at least one agent* A_i. *The agents want to agree on instantiations such that all the constraints are satisfied.*

Definition 4 *The set of modifiers of* x_i, M_i^s, *is the set of agents* A_i *having* x_i *in their* M_i.

$$M_i^s = \{A_k | x_i \in M_k\}$$

3.1. General Replica-based MAS

Replica-based Multiply Asynchronous Search (R-MAS) is a technique based on a special DisCSP-based modeling of the problem (besides unifying ABT, ABTR, AAS, and DMAC-ABT). Namely a set of distinct virtual agents (each enforcing another hierarchical level of abstraction, i.e. relaxations), replace together each physical agent's problem. This model is then solved with some known algorithm and a set of minor implementation optimizations. These optimizations are obtained by taking into account that several of the agents in this new problem are actually facets of one and the same agent and can share messages and data structures (agent views, consistency-levels).

Multiply Asynchronous Search (MAS) is the algorithm that integrates asynchronous search, ABT, the reordering of ABTR, the consistency maintenance of DMAC, and the aggregations of AAS [37, 39, 40, 35]. R-MAS maintains a (dynamic) total order on the virtual agents involved in computation. All the agents having interests (constraints, nogoods) on a variable x in M_i^s are inserted into a list maintained for x by A_i, called *outgoing-links*.

Remark 1 *MAS can work with most modelings of a DisCSP, even if it works best when levels of abstraction are introduced such that each agent is modifier of at most one variable. An agent that is modifier of several variables can be easily decomposed in several abstract (aka virtual) agents such that this condition is satisfied [47]. Note that some agents may not be modifiers for any variables.*

Agents exchange **ok?**, **add-link**, and **propagate** or **nogood** messages with aggregates, interests, respectively nogoods, according to the standard usage of these messages [46, 35]. Abstracting from all details, the following concepts are used here.

Definition 5 *An* **aggregate** *(*assignment*) is a triplet* $\langle x_j, s_j, h_j \rangle$ *where* x_j *is a variable,* s_j *a set of values for* x_j, $s_j \neq \emptyset$, *and* h_j *a signature of the pair* (x_j, s_j).

Agents propose aggregates to restrict the possible values of variables in different contexts of the exploration of the search space. The signature helps to guarantee a correct message ordering. It determines if a given aggregate is more recent than another. A **signature** is a chain h of pairs, |a:b|, that can be associated to a proposal, Z, for a variable x. A pair p=|a:b| in h signals that Z complies to the b^{th} proposal for x that was made by the agent with position a. A total order, *stronger*, is induced on signatures by the

compliance with recent proposals of higher priority agents. A proposal made by an agent A_i is valid if no newer known proposal was built by agents that are not ordered after A_i in the current total order on agents.

3.2 DMAC-ABT

We recall (see any basic AI manual like Russell&Norvig's [33]) that local arc/bound consistency is a technique of relabeling variables with more precise domains. A *label* for x_i is nothing else than a set of values containing all the valuations of x_i appearing in some solution. In most initial problem descriptions variables have in their domains values that cannot appear in any solution. While such values add exponential complexity to systematic search techniques, some of them can be detected and eliminated with local observations on small subproblems, eliminations that only require polynomial effort. This is why recalculation/shrinking of labels is a principled technique that is very recommended, specially in its forms whose cost complexity is of a low polynomial order (node, arc, bound, singleton consistencies).

Maintaining Asynchronously Consistencies in asynchronous search (DMAC-ABT) is proposed in [40, 35]. The algorithm consist in running ABT, on top of which distributed 'local' consistency achievement is enforced independently and concurrently for each subproblem generated by the the last proposed assignments of agents $A_i, i < k$. Each of the n subproblems are obtained by taking for k a distinct value in $\{1..n\}$.

The algorithm discussed in detail in [40, 35] is repeated in Algorithms 1, 2, and 3.

```
when received (ok?,⟨x_j, d_j, , c_{x_j}⟩) do
    if(old c_{x_j}) then return;
1.1 add(x_j,d_j,c_{x_j}) to agent view;
    eliminate invalidated nogoods;
    maintain_consistency(j);
    check_agent_view; //only satisfies consistency nogoods of levels t, t<cL_i;
end do.
procedure check_agent_view do
    when agent view and current_value are not consistent
    //cf. nogoods of levels t, t<cL_i
        if no value in D_i is consistent with agent view then
            backtrack;
        else
            select d ∈ D_i where agent view and d are consistent;
            current_value ← d;
            C^i_{x_i} ++;
            maintain_consistency(i);
            send (ok?,⟨x_i, d, C^i_{x_i}⟩) to lower priority agents in outgoing links;
        end
end do.
```

Algorithm 1: *Procedures of A_i for receiving* **ok?** *messages in DMAC-ABT.*

4. R-MAS for DisWCSPs

To use R-MAS with DisWCSPs, the idea is to model the value of a tuple in a proposal with a new variable whose domain is $\mathbb{R}$.

4.1. Optional Aggregations

Using aggregations is optional but interesting for privacy reasons. Aggregation is just a generalization of the case where a tuple is taken at a time. For correct evaluation of the value of a proposal in an aggregate, the proposed aggregates have to be built in such a way that the local cost is identical for all tuples of the known partial valuation (namely in the intersection of this proposal with all known valid proposals).

When constraints have tuples with very non-uniform values, one can still exploit wide aggregations by employing hierarchical abstractions. This can be done efficiently upon the technique of R-MAS, by allowing splitting of constraints (which is an abstraction technique) in such a way that all/several tuples in an abstract agent can be aggregated (see Figure 1). The only requirement is that the sum of the cost of a tuple in the obtained constraints equals the initial cost.

This splitting can be done in a greedy way, similar to [38]. Alternatively, many clustering techniques can be straightforwardly used to get such splitting [29]. This technique is optional and we will therefore not discuss it here to avoid over-burdening the reader. It is nevertheless introduced in notations to show that the description given extends with no modification to cases where splitting for aggregation will be used in the future.

4.2. Branch and Bound with cost variables

Let us therefore introduce a new variable x_{c_i}, $x_{c_i} \geq 0$ for each agent A_i. These variables model the cost of the current proposal, the value of f_i, which should be the same for all valuations in the set of aggregates currently known by A_i. Since all agents are interested in the variables x_{c_i}, all the agents are in the *outgoing-links* of each agent A_i for the variable x_{c_i}. A_i proposes $x_{c_i}=\{k\}$ when he proposes an aggregate whose local cost is k (e.g. k=4 when the proposal is $(x_1 \in \{0..2\}, x_2 \in \{1..3\})$ for the first virtual agent obtained by splitting in Figure 1).

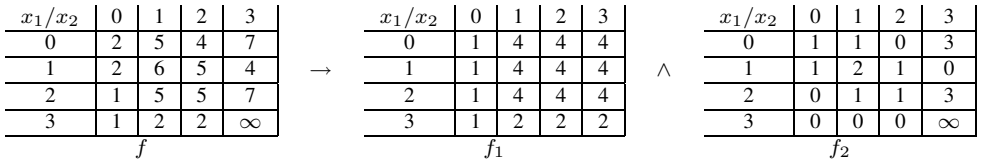

Figure 1. Example of constraint splitting for exploiting aggregations with distributed weighted CSPs. A weighted constraint between x_1 and x_2 with domains in $\{0..3\}$ is splitted in two other constraints. Two virtual agents replace the original one with R-MAS.

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 $\neg N$;
    end
  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 all new assignments in $\neg N$ to *agent_view*;
2.1 reconsider stored and invalidated nogoods;
  end
  old_value $\leftarrow$ *current_value*; **check_agent_view**;
  when old_value = current_value
2.2 send (**ok?**,$\langle x_i$, *current_value*, $C^i_{x_i}\rangle$) to A_j;

end do.
procedure backtrack do
  nogoods $\leftarrow \{V | V =$ 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;
2.3 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;
  end do
  check_agent_view;
end do.

Algorithm 2: *Procedures of A_i for receiving* **nogood** *messages in DMAC-ABT.*

In Branch&Bound the idea is to discard search paths for which it is proven that any enclosed solution is more expensive than any already found solution. Any solution with value C defines therefore a *nogood* (i.e. dynamically inferred constraint), $\sum_i x_{c_i} < C$, that is broadcast to all agents. It is known that $x_{c_i} \geq 0$ therefore each agent can enforce the weaker constraint:

$$\sum_{\text{known } x_{c_i}} x_{c_i} < C$$

No other modification is required and a new Branch and Bound algorithm is obtained. The last found solution is optimal. This algorithm is called R-MAS-BB-c1.

Remark 2 *With R-MAS-BB-c1, the value of a solution is given by the sum of the values assigned in it to the x_{c_i} variables.*

Solutions are detected in MAS according to the algorithm described in [39, 35]. Each time that a solution is detected by the broker, a **solution** message is broadcasted to participants with the value C of the obtained solution. Algorithm 4 shows how the constraint $\sum_i x_{c_i} < C$ is added to each agent.

Example 1 *An* **ok?** *sent by the abstract agent of the first constraint obtained in Figure 1 can be:* **ok?**$(\langle x_1, \{0\}, |1{:}0|\rangle \langle x_2, \{1..3\}, |1{:}0|\rangle \langle x_{c_1}, \{4\}, |1{:}0|\rangle)$. *This specifies that the R-MAS 'virtual' agent enforcing the constraint f_1 agrees to x_1=0, x_2 being any of the values $\{1..3\}$, and x_{c_1}=4.*

Proposition 1 *R-MAS-BB-c1 is correct, complet, terminates, and finds the optimal solution.*

Proof. *The proof is immediate from the correctness of R-MAS and by construction (introduction of Branch&Bound which is known to be correct).*

4.3. Cost of nogoods (WR-MAS)

In the previous section it can be noticed that cost conflicts are only detected from partial valuations. A better idea has been introduced for centralized techniques

when received(**propagate**,A_j,k,$c^k_{x_v}(j)$,$V\rightarrow(x_v\notin l)$) **do**

3.1 **when** have higher tag $c^k_{x_v}(j,i)\geq c^k_{x_v}(j)$ **then** return;
$c^k_{x_v}(j,i) \leftarrow c^k_{x_v}(j)$;
when any $\langle x,d,c\rangle$ in V is invalid (old c) **then** return;
when $\langle x_u,d_u,c_u\rangle$, *where* x_u *is not connected, is contained in V*
send **add-link** to A_u;
add $\langle x_u,d_u,c_u\rangle$ to *agent view*;

3.2 add other new assignments in V to *agent view*;
eliminate invalidated nogoods;
$cn^k_{x_v}(i,j) \leftarrow \{V\rightarrow(x_v\notin l)\}$;
maintain_consistency(minimal level that is modified);
check_agent_view; //only satisfies consistency nogoods of levels t, t$\leq$cL$_i$;
end do.
procedure maintain_consistency(*minT*) **do**
if (minT $>$ cL$_i$) **then** return;

3.3 **for** (*t*$\leftarrow$*minT; t*$\leq$*i; t++*) **do**
new-cns $\leftarrow$ consistency nogoods for $x_k\in$vars(A_i) after local consistency on $P_i(t)$;
when (*domain wipe out by computing explicit nogoods* nogoods)
for every $V\in$ nogoods **do**
select $\langle x_j,d_j,c_{x_j}\rangle$ where x_j has the lowest priority in V;

3.4 send (**nogood**,A_i,V) to A_j;
eliminate invalidated explicit nogoods;
cL$_i$ $\leftarrow$t;
remove $\langle x_j,d_j,c_{x_j}\rangle$ from *agent view*;
end do
break;
for every *new-*c_{x_u} (*consistency nogood for* x_u) $\in$ new-cns **do**
when *new-*c_{x_u} *shrinks label of* x_u (*obtained from* $\cup_{w,k\leq t} cn^k_{x_u}(i,w)$)
$cn^t_{x_u}(i,i) \leftarrow$ *new-*c_{x_u};
$c^t_{x_u}(i)$++;
send (**propagate**,A_i,t,$c^t_{x_u}$,*new-cn*) to interested agents $A_j, j\geq t$;
end do
end
end do.

Algorithm 3: *Procedure of* A_i *for receiving* **propagate** *messages in DMAC-ABT1.*

in [21, 22, 30]. They explain how cost of subproblems can be computed by consistency propagation for estimating bounds earlier: *Use the cost of a constraint only once.*

In order to apply the previous techniques to R-MAS, we redefine the notion of consistency nogoods as follows.

Definition 6 (SRC) *A set of references to constraints (SRC), is a set of symbols (e.g.* $\{C_{f_3},C_{f_5},C_{f_7}\}$*), where* C_{f_i} *is a reference to the constraint,* f_i*, of the DisWCSP.*

when received (**solution**,C) **do**
add f(x) to the set of local constraints:

$$f(x)=\begin{cases}\infty & \text{if } \sum_{\text{known } x_{c_i}} x_{c_i}\geq C\\ 0 & \text{if } \sum_{\text{known } x_{c_i}} x_{c_i}<C\end{cases}$$

end do.
procedure solution-detected (*solution*) **do**
$C \leftarrow \sum_{(x_{c_i},C_i,k_i)\in solution} C_i$;
broadcast (**solution**,C)
end do.

Algorithm 4: Procedure of A_i for receiving **solution** messages in R-MAS-BB-c1.All other procedures are inhereted from DMAC-ABT. The procedure *solution-detected* is run by whoever detects and builds the solution (e.g. broker). If each agent builds the solution separately then the message needs not be broadcasted but just delivered locally.

Remark 3 *There is no need to attach a* reference to constraint *to hard constraints like* $\sum_{known\ x_{c_i}} x_{c_i}<C$ *since there is no problem in applying them redundantly:* $\infty+\infty=\infty$*, and* $0+0=0$.

Definition 7 (Weighted Consistency nogood) *A weighted consistency* nogood *(WCN) for a level (i.e. search depth)* k *and a variable* x *has either the form* $\langle srcs,c_1,c_2,V\cup(x\in l^k_x)\rangle$ *or* $\langle srcs,c_1,c_2,V\cup\neg(x\in s\backslash l^k_x)\rangle$.
V *is a set of assignments. Any assignment in* V *must have been proposed by* A_k *or its predecessors.* l^k_x *is a label,* $l^k_x\neq\emptyset$. $srcs$ *is a set of references to constraints while* c_1 *and* c_2 *are low bounds of the cost of the constraints referred by* $srcs$ *given* V *and values remaining, respectively values eliminated for* x *by* l^k_x. s *is the initial domain of* x.

Remark 4 (Hard WCNs) *Most often* c_2 *will be* ∞*, therefore we will often use for WCNs the simplified notation* $\langle srcs,c_1,V\cup(x\in l^k_x)\rangle$ *that implies* $c_2=\infty$.

Example 2 *Take as example the WCN* $\langle\{C_{f_3},C_{f_5}\},27,\infty,(\langle x_2,\{1..3\},|1{:}0|\rangle\cup(x_4\in\{3..5\}))\rangle$. *This nogood states that as long as the assignment* $\langle x_2,\{1..3\},|1{:}0|\rangle$ *is valid, the sum of the values due to the constraints referenced by* C_{f_3},C_{f_5} *and unspecified hard nogoods is low bounded by 27 when* $x_4\in\{3..5\}$*, respectively low bounded by* $+\infty$ *(i.e. infeasible) otherwise.*

The new concepts are the basis of a new family of asynchronous algorithms that extend R-MAS. We call the new family: Weighted R-MAS (WR-MAS).[2]

[2] In [35] these are called VR-MAS respectively DVR-MAS.

Several WCNs can be stored by an agent for a variable at different depths in the search. A delicate problem is the combination of WCNs. Two consistency nogoods that can be combined in R-MAS can also be combined in WR-MAS in their weighted form.

Definition 8 (valid weighted nogood) *A weighted consistency nogood is valid only as long as all the aggregates involved in it are valid.*

4.4. Inference with weighted consistency nogoods

Proposition 2 *Any two weighted consistency nogoods,* $\langle src_1, c_1, c'_1, N_1 \cup x \in l_1 \rangle$ *and* $\langle src_2, c_2, c'_2, N_2 \cup x \in l_2 \rangle$ *where any aggregates in* N_1 *and* N_2 *for the same variable do not invalidate each other, can be combined into a new weighted consistency nogood. The obtained nogood is* $\langle src, c, c', N \cup x \in l \rangle$ *such that* $src = src_1 \cup src_2$, $c = max(c_1, c_2)$, $c' = min(c'_1, c'_2)$, $l = l_1 \cap l_2$, *and* $N = N_1 \cup N_2$, N *retaining only the strongest among two aggregates for the same variable.*

The operator combining two WCNs is denoted $\oplus$.

Example 3 *Consider first the general case:*
$(\langle \{C_{f_1}, C_{f_2}\}, 27, \infty, x_1 \in \{2..5\} \rangle \oplus \langle \{C_{f_2}, C_{f_3}\}, 15, 1000, x_2 \in \{4..7\} \rangle) \rightarrow \langle \{C_{f_1}, C_{f_2}, C_{f_3}\}, 27, 1000, x_2 \in \{4, 5\} \rangle)$
For WCNs that have a single cost (see Remark 4):
$(\langle \{C_{f_1}, C_{f_2}\}, 27, x_1 \in \{2..5\} \rangle \oplus \langle \{C_{f_2}, C_{f_3}\}, 15, x_2 \in \{4..7\} \rangle) \rightarrow \langle \{C_{f_1}, C_{f_2}, C_{f_3}\}, 27, x_2 \in \{4, 5\} \rangle)$

Remark 5 *A stronger WCN can be computed in Proposition 2 by taking:* $c' = min(max(c'_1, c_2), max(c_1, c'_2), max(c'_1, c'_2))$. *It should be remarked that the semantic of removed values and remaining values can be exchanged, and the 4 possible combinations lead to 4 distinct inferences.*

Corollary 2.1 *Any two weighted consistency nogoods,* $\langle src_1, c_1, c'_1, N_1 \cup x \in l_1 \rangle$ *and* $\langle src_2, c_2, c'_2, N_2 \cup x \in l_2 \rangle$ *where any aggregates in* N_1 *and* N_2 *for the same variable do not invalidate each other, can be combined into a new weighted consistency nogood. The obtained nogood is* $\langle src, c, c', N \cup x \in l \rangle$ *such that* $src = src_1 \cup src_2$, $c = min(c_1, c_2)$, $c' = max(c'_1, c'_2)$, $l = l_1 \cup l_2$, *and* $N = N_1 \cup N_2$, N *retaining only the strongest among two aggregates for the same variable.*

Proposition 3 *When* $src_1 \cap src_2 = \emptyset$ *in Proposition 2, the estimation can be tighter:* $c = c_1 + c_2$, $c' = min(c'_1 + c_2, c'_2 + c_1, c'_1 + c'_2)$.

Example 4 *Consider first the general case:*
$(\langle \{C_{f_1}, C_{f_2}\}, 27, \infty, x_1 \in \{2..5\} \rangle \vee \langle \{C_{f_3}, C_{f_4}\}, 15, 1000, x_2 \in \{4..7\} \rangle) \rightarrow \langle \{C_{f_1}, C_{f_2}, C_{f_3}, C_{f_4}\}, 42, 1027, x_2 \in \{4, 5\} \rangle)$
For WCNs that have a single cost (see Remark 4):
$(\langle \{C_{f_1}, C_{f_2}\}, 27, x_1 \in \{2..5\} \rangle \vee \langle \{C_{f_3}, C_{f_4}\}, 15, x_2 \in \{4..7\} \rangle) \rightarrow \langle \{C_{f_1}, C_{f_2}, C_{f_3}, C_{f_4}\}, 42, x_2 \in \{4, 5\} \rangle)$

The $\oplus$ operator will denote in the following the operator that combines two WCNs to the tightest WCN.

4.5. HOWTO infer WCNs

Let us now see how agents can infer WCNs and how WCNs are propagated.

Generating/Strengthening WCNs Given a set N of valid assignments known by A_i and a set M of hard valid WCNs at search depth h, let T be the set of tuples allowed by them in the k-ary constraint f_i. We can infer k WCNs: $\langle srcs, c, V \cup (x_{ij} \in l^h_{ij}) \rangle$, $j \in [1, k]$. Here:

- $srcs$ is the union of the SRCs of the WCNs in M, and also contains the reference to f_i.
- Let c_M be the maximum cost that can be obtained combining costs of WCNs. Costs are summed for WCNs having disjoint SRCs and the maximum is taken among costs of WCNs whose SRCs are not disjoint. Different order of applying these two operations lead to different results and backtracking is needed to search the order leading to the highest c_M.

  Let c_T be the minimum value that f_i attaches to a tuple in T. If the SRC of f_i is not in the SRCs found in M, then $c = c_T + c_M$, otherwise $c = max(c_T, c_M)$.
- V is the union of all the assignments in M and N.
- x_{ij} is the j^{th} variable involved in the k-ary constraint f_i.
- l^h_{ij} is the label resulting at search depth h for x_{ij} after applying the proposals and WCNs in M and N.

Algorithm 5 shows in details the generation and propagation of WCNs.

Example 5 *We will consider the virtual agent* A_3 *enforcing the constraint* f_1 *in Figure 1. If* A_3 *knows an assignment* $x_2 \in \{1, 2\}$ *|1:2| then it can infer the WCNs* $\langle \{C_{f_1}\}, 2, \langle x_2, \{1, 2\}, |1{:}2| \rangle \cup (x_2 \in \{1, 2\}) \rangle$ *and* $\langle \{C_{f_1}\}, 2, \langle x_2, \{1, 2\}, |1{:}2| \rangle \cup (x_1 \in \{0..3\}) \rangle$

Propagating WCNs A value v in the label of variable x is removed if the addition of an assignment $x \in \{v\}$ can lead to the inference of a WCN that together with the known constraints and assignments leads to an explicit nogood. A WCN $\langle srcs, c, L \rangle$ known by an agent A_i leads to an explicit

procedure Generate *(labels,WCSP,nogoods,agent-view)* **do**
  wcns← ∅;
  $c_T \leftarrow$ minimum cost of WCSP for a tuple in labels;
  for all *x∈variables(WCSP)* **do**
    swcns←selected wcns in nogoods;
    $(srcs, c_M, V \cup l) \leftarrow$combination with $oplus$ of swcns and agent-view;
    if *SRC(WCSP)∈srcs* **then**
      wcn← $(srcs, max(c_T, c_M), V \cup l)$;
    else
      wcn← $(srcs \cup \{SRC(WCSP)\}, c_T + c_M, V \cup l)$;
    end
    $wcns \leftarrow (srcs, c_M, V \cup l)$;
  end do
  return wcns;
end do.
procedure Infeasible *(wcn, agent-view)* **do**
  cost←cost(wcn);
  for all $x_{c_i} \in$*agent-view* **do**
    if $x_{c_i} \notin srcs(wcn)$ **then**
      cost←cost+value(x_{c_i},agent-view);
    end
  end do
  if *know constraint* $\sum_{known} x_{c_i} x_{c_i} < C$, $C \leq$ *cost* **then**
    return true;
  end
  return false;
end do.
procedure Filter *(variable, labels, WCSP)* **do**
  label←labels[variable];
  for all $v \in$ *label* **do**
    newlabels←labels;
    newlabels[variable]←$\{v\}$;
    wcns←Generate(newlabels,WCSP,nogoods,agent-view);
    if *(InFeasible(wcn,agent-view) for some wcn∈wcns)* **then**
      remove v from labels[variable];
      post the other variables for filtering;
    end
  end do
end do.

Algorithm 5: Weighted Distributed Arc/Bound Consistency filtering algorithm

nogood if A_i knows a constraint $\sum_{\text{known}} x_{c_i} x_{c_i} < C$, and it knows assignments for a set K of cost variables not in $srcs$ such that $sum_{(x_{c_k}, C_k) \in K} C_k + c \geq C$.

Example 6 *Consider that in Example 5, A_1 learns the nogood $\sum_{known} x_{c_i} x_{c_i} < 7$ from a message* **solution***(7) and receives the proposal $\langle x_{c_3}, \{5\}, |2{:}3| \rangle$. Each of the bounds $b \in \{0,1,2\}$ of x_1 fail successively if an assignment $(x_1 \in \{b\})$ is added to the knowledge of A_1. Therefore A_1 obtains by propagation the WCN:*
$\langle \{C_{f_1}\}, 2, \langle x_2, \{1,2\}, |1{:}2| \rangle \langle x_{c_3}, \{5\}, |2{:}3| \rangle (x_1 \in \{3\}) \rangle$

4.6. Data Structures for WR-MAS

The family of algorithms proposed here, Weighted Replica-based Multiply Asynchronous Search (WR-MAS), builds on R-MAS by adding the use of WCNs in the consistency maintenance.

The following approaches are known for maintaining data structures with nogood-based consistency (considering that labels are treated as ranges):

- DMAC0: Storing at most the last valid consistency nogood (CN) per variable (related to what was done in [19] for each value).
- DMAC1: Storing at most the last valid CN per variable per search depth (as in MHDC [35]).
- DMAC2: Storing at most the last valid CN per variable per search depth per agent generating CNs (as in the version of DMAC published in [40]).
- DMAC3: Storing at most the last valid CN per variable per search depth per agent generating CNs and per agent whose constraints are not involved in the CN (as in [35] for robustness in treating openness).

All of the previous four alternatives translate and work straightforwardly with WR-MAS, where one just uses WCNs instead CNs. The resulting techniques are therefore called: WDMAC0, WDMAC1, WDMAC2, WDMAC3.

It is reasonable to expect that an important new alternative that becomes reasonable (for problems without openness or in order to allow oneself the highest efficiency in combining WCNs) is to:

- WDMAC4: Store at most the last valid WCN:
  - per variable ($\times$ m variables),
  - per search depth ($\times a$ virtual agents),
  - per agent generating WCNs ($\times a$ virtual agents),
  - and per combination of involved SRCs (up to k, $k \geq 2^c$, combinations may be used).

The memory requirement for the new version is:

$$O(ma^2 2^c(md+c)) \quad (1)$$

where m is the number of shared variables, a is the number of virtual agents ($\geq n$), c is the number of constraint references (=a), and d is the maximal domain size.

It can be noted that these requirements are exponential in the number of constraint references (number of weighted constraints). To meet space constraints, a fix subset of size k of combinations of constraint references has to be chosen. The corresponding space complexity becomes $O(ma^2k(md+c))$ which is now polynomial.

Remark 6 *The proof of DMAC [40] that the highest achievable degree of consistency is achieved at quiescence applies to the new strategy when the storage of the strongest computed WCNs for each generating agent, variable and search depth is also guaranteed and all agents store WCNs for the same set of k constraint references.*

4.7. Backtrack in Extended Branch and Bound (WR-MAS algorithms)

We have already presented almost all features on a new family of algorithms, Weighted Replica-based MAS, and we have also seen in detail a particular algorithm of this family, namely DMAC-ABT. It remains us to explicit the way in which backtracking builds and sends explicit nogood messages.

Weighted Replica-based MAS uses the previously mentioned form for WCNs. The cost C of any solution is broadcast under the form of a nogood

$$\sum_i x_{c_i} < C,$$

as in R-MAS-BB-c1 (it could be similarly based on R-MAS-BB-c2 [35]).

WR-MAS starts with all agents enforcing a constraint $\sum_i x_{c_i} < \infty$, and the constraints $x_{c_i} \geq 0$. They build and propose aggregates to all lower priority agents in corresponding outgoing-links. An agent builds a (hard) explicit nogood in any of the following cases (see Algorithm 6):

- It knows a constraint $\sum_i x_{c_i} < C$, and valid assignments to a set S of cost variables, such that $\sum_{i \in S} x_{c_i} \geq C$.
- It knows a WCN $\langle src, c, M \rangle$, a constraint $\sum_i x_{c_i} < C$, and valid assignments to a set S of cost variables ($S \cap src = \emptyset$), such that $c + \sum_{i \in S} x_{c_i} \geq C$.
- When it exhausts its search space and still cannot generate any new proposal, the agent generates an (eventually optimized) explicit nogood. The new explicit nogood is composed of valid received proposals by combining the nogoods entailed by the received valid proposals with the explicit nogoods previously received for its own proposals and that helped exhausting the current local search space.

Remark 7 *An agent will not send proposals with costs that he can himself combine with other costs, about which he was told, to infer explicit nogoods. When the set of proposals an agent knows changes, it verifies whether he can infer nogoods for its current proposals and abandons them if it succeeds.*

when received(propagate, A_j,k,$c^k_{x_v}(j)$,*SRC*,*c*,*V*$\rightarrow(x_v \notin l)$) **do**

5.1 **when** have higher tag $c^k_{x_v}(j,i) \geq c^k_{x_v}(j)$ **then** return;
$c^k_{x_v}(j,i) \leftarrow c^k_{x_v}(j)$;
when any $\langle x, d, c \rangle$ in V is invalid (old c) **then** return;
when $\langle x_u, d_u, c_u \rangle$, *where x_u is not connected, is contained in V*
send **add-link** to A_u;
add $\langle x_u, d_u, c_u \rangle$ to *agent view*;
add other new assignments in V to *agent view*;
eliminate invalidated nogoods;
$cn^k_{x_v}(i,j) \leftarrow \{SRC,c,V\rightarrow(x_v \notin$l$)\}$;
maintain_consistency(minimal level that is modified);
check_agent_view; //only satisfies consistency nogoods of levels t, t$\leq$cL$_i$;
end do.
procedure backtrack do
nogoods $\leftarrow \{V | V =$the set of *agent_view* involved in an inference described in Section 4.7$\}$;
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;
5.2 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 (and WCNs);
end do
check_agent_view;
end do.

Algorithm 6: Procedure `backtrack` and receiving **propagate** messages for agent A_i in WDMAC-ABT.

4.8. Weighted Bound Consistency

Arc consistency based on values has already been used many times with weighted CSPs. One of the most famous algorithms is given in [22]). In difference to [22, 30] we do not associate constraints to variables having to ensure that each constraint is counted only once by tricks in constraint representation, but rather we associate each weighted constraint with a constraint reference, keeping track of which constraints are involved in which costs.

Based on (hard) WCNs, one can simply use ranges and Bound Consistency as in most discussed implementations of DMAC and as in [30]:

As shown, a new weighted consistency nogood can be generated by proving that a certain bound of a variable leads to local cost that together with the view and nogoods involved in the computation lead to a conflict against a constraint $\sum_i x_{c_i} < C$.

5. Adopt±PFC-MRDAC

In [36] we explained that there are two key elements that are required for improving efficiency for optimization with large problems:

- Limiting commitment. This consists in abandoning a branch if it is not promising.
- Using acceptable value ordering heuristics.

Our motivation in proposing these two techniques was that expensive paths should be abandoned without fully exploring them.

The question with *limiting commitment* is how to decide when a branch should be abandoned. A short timeout would not scale with the problem. Recent research [27] returns to a more classic and principled alternative, namely to abandon commitments according to the A^* heuristic. Whenever the estimated cost of another branch looks more promising, the current commitment can be broken.

Remark 8 *To avoid that two much work is lost too often by discarding nogoods due to frequently abandoning commitments (phenomena often encountered in ABTR-wc), a common solution is to abandon only when the heuristic has a higher confidence (e.g. the estimated cost of the current branch is (1+k) times more expensive than the estimation for the best alternative).*

To notice that the main theoretic result of A^* applies, namely:

Remark 9 *If the estimation of the cost of a path is either perfect or optimistic and if $k = 0$, then the first reached solution is the optimal one.*

Adopt provides information on low bounds by introducing a new type of messages in ABT, messages that transport current estimates toward predecessor agents. This limited ordering flexibility. We exploit in the following a stronger mechanism offered by consistency maintenance with (hard) WCNs.

An optimistic estimate (low bound) of the cost of a branch can be obtained in WR-MAS (and WDMAC-ABT) as follows.

Remark 10 *For an agent A^i, a tight conservative (optimistic) evaluation of the cost of a branch B that it can propose is given by the sum between the cost of view(A^i), $\sum_{c_j \in view(A^i)} x_{c_j}$, and the highest cost c of a WCN, $(SRC, c, N \cup B \cup (x \in l))$, that can be inferred.*

Remark 11 *We assume here that we use the version of DMAC-ABT where the agent generating a proposal receives all the (weighted) CNs at the level it generates [41].*

Remark 12 *The technique of using backtrack nogoods in consistency [40] is the one guaranteeing that the feedback obtained in the new algorithm is strictly stronger than in Adopt.*

The following observation can be applied in using more WCNs to estimate costs: $\forall SRC, c \in \mathbb{R}, N, l$, where SRC is a set of references to constraints, c is their lower bound cost given a view N and a label l for x, if $(SRC, c, N \cup x \in l')$ is a WCN, then one can infer $(SRC, c, N \cup x \in l')$, $\forall l' \subset l$.

There is a straightforward way to implement a value ordering heuristic in WDMAC-ABT based on the low bounds (also used in Adopt, see [27]). Namely, the value with the lowest low bound is chosen first.

Remark 13 *When the commitment on a branch is abandoned, the value ordering heuristic to be used in agreement with A^* is to propose the assignment (aggregate-set) that has the lowest estimated cost, within the current level of abstraction of the agent/replica.*

The algorithm obtain with this extension to DMAC-ABT is called Adopt-PFC-MRDAC (and the extension of the family of algorithms WR-MAS is called DWR-MAS[3]).

6. Interesting Extensions

Several possible improvements and extensions are possible. We will only shortly mention one of them.

[3] In [35] we presented it as DVR-MAS.

when received(propagate,A_j,k,$c^k_{x_v}(j)$,*SRC,c,V→(*$x_v \notin$*l))* **do**

6.1 **when** have higher tag $c^k_{x_v}(j,i) \geq c^k_{x_v}(j)$ **then** return;
$c^k_{x_v}(j,i) \leftarrow c^k_{x_v}(j)$;
when any $\langle x, d, c\rangle$ in V is invalid (old c) **then** return;
when $\langle x_u, d_u, c_u\rangle$, *where* x_u *is not connected, is contained in V*
send **add-link** to A_u;
add $\langle x_u, d_u, c_u\rangle$ to *agent view*;

6.2 add other new assignments in V to *agent view*;
eliminate invalidated nogoods;
$cn^k_{x_v}(i,j) \leftarrow$ {SRC,c,V→($x_v \notin$l)};
maintain_consistency(minimal level that is modified);
reestimate low bounds //using new WCNs and $x_{c_k}, \forall k$
check_agent_view; //only satisfies consistency nogoods of levels t, t≤cL$_i$;

end do.

procedure check_agent_view do
when agent view *and* current_value *are not consistent or when have value with better loow bound //cf. nogoods of levels t, t<cL$_i$*
if *no value in* D_i *is consistent with* agent view **then**
backtrack;
else
select $d \in D_i$ with lowest low bound where *agent view* and d are consistent;
current_value ← d;
$C^i_{x_i}$++;
maintain_consistency(i);
send (**ok?**,$\langle x_i, d, C^i_{x_i}\rangle$) to lower priority agents in *outgoing links*;
end
end do.

Algorithm 7: Procedure of A_i for receiving **propagate** messages in Adopt-PFC-MRDAC. All the other procedures are inherited from WDMAC-ABT.

6.1. Weighted Arc Consistency

There are several ways to extend WR-MAS to arc consistency:

1. Modify the definition of hard WCNs such that each value in the label is associated with a different cost.
2. Going even further than the previous alternative and associate each value with a distinct SRC and cost.
3. The same expressive power as in the previous alternative can be reached by allowing the use of WCNs in their most general defined form, namely where the cost of values that are not contained in label is not ∞.

The usage of WCNs $\langle srcs, c, c', M\rangle$ where $c' \neq \infty$ can help to concentrate the reasoning on some values in the label. This can help to prune certain particularly expensive regions inside the bounds, even if one does not want a full arc-consistency.

6.2. Related Work

[27, 35, 23, 11] describe/recapitulate most of the work on asynchronous optimization pursued during the last few years for branch and bound over weighted DisCSPs. Several other synchronous or centralized techniques have been described by many authors. An algorithm mentioning for the first time the A* heuristic in asynchronous distributed optimization appears in [27]. Several research groups are active in the DisCSP and soft CSP communities [8, 14, 46, 7, 42, 23, 34, 26, 16, 17, 28, 11, 43, 6, 15, 48, 2, 20, 9, 25, 4, 24, 12, 10, 3, 45, 18, 1, 32, 31, 44, 13]. Many other existing algorithms for DisCSPs can be adapted for optimization.

The main reason for using DisCSPs lies in the privacy that can be offered to agents is this framework. Alternative ways of approaching privacy requirements are described and compared in [35] against constructive search for DisCSPs. The advantage of the last ones consists in some additional efficiency and in more control of an agent over its privacy loss.

7. Conclusions

Distributed optimization is an expensive task. Initially authors tried different types of hill-climbing and approximate techniques [23]. Several other synchronous Branch&Bound and A* techniques appeared last decade mainly in work reported by Dr. Yokoo's team. In [36] we proposed an asynchronous Branch&Bound technique that is based on ABT/AAS. Preliminary tests that we performed at that time have shown the technique to be prohibitivly expensive on a simple real-world problem. Recently, another

interesting technique, called Adopt, shows how A* value ordering heuristic can be introduced in ABT. While it is not yet known how the two existing asynchronous optimization techniques compare (namely Adopt vs ABT/AAS with Branch&Bound), here we have shown how these two techniques can be combined.

The main new ideas proposed in this article are that:

1. Consistency achievement or maintenance in Weighted DisCSPs can be performed if the Consistency Nogood concept of DMAC-ABT is enriched to a more general concept: The Weighted Consistency Nogood (WCN). We prove rules of inference with WCNs.

2. An asynchronous equivalent of the best available centralized technique, PFC-MRDAC, is obtained by mixing the aforementioned consistency maintenance with Branch&Bound.

3. The feedback that Adopt needs about low bounds on constraints of successor agents, can be extracted using cost attached to labels in WCNs and detected by the previously mentioned 'local' consistency process.

In this paper we outlined the steps required for *asynchronizing* PFC-MRDAC for Distributed Weighted CSPs and we shown how consistency maintenance can be added to Adopt. More exactly PFC-MRDAC is obtained for certain synchronization and agent strategy in WR-MAS. DWR-MAS is nevertheless much more general and can lead to a very different behavior depending on network timing and agent strategies. The evaluation of different versions of WDMAC-ABT and Adopt-PFC-MRDAC is one of our immediate plans.

References

[1] F. Arbab and E. Monfroy. Distributed splitting of constraint satisfaction problems. In *Coordination 2000*, pages 115–132, 2000.

[2] A. Armstrong and E. F. Durfee. Dynamic prioritization of complex agents in distributed constraint satisfaction problems. In *Proceedings of 15th IJCAI*, 1997.

[3] M. Barbuceanu and W.-K. Lo. A multi-attribute utility theoretic negotiation architecture for electronic commerce. In *Proc. of AA'2000*, pages 239–246, 2000.

[4] B. Baudot and Y. Deville. Analysis of distributed arc-consistency algorithms. Technical Report RR-97-07, U. Catholique Louvain, 1997.

[5] F. Benhamou and F. Goualard. Universally quantified interval constraints. In *Procs. of CP'2000*, pages 67–82, 2000.

[6] M. Berry, Pauline, Y. Choueiry, Berthe, and L. Friha. A multi-agent architecture for a distributed approach to resource allocation using temporal abstractions. Technical Report TR-92/18, EPFL, 1992.

[7] C. Bessière, A. Maestre, and P. Meseguer. Distributed dynamic backtracking. In *CP*, page 772, 2001.

[8] S. Bistorelli, U. Montanari, and F. Rossi. Constraint solving over semirings. In *Proceedings IJCAI*, pages 624–630, Montreal, 1995.

[9] Z. Collin, R. Dechter, and S. Katz. On the feasibility of distributed constraint satisfaction. In *Proceedings of IJCAI 1991*, pages 318–324, 1991.

[10] S. E. Conry, K. Kuwabara, and V. R. Lesser. Multistage negotiation for distributed constraint satisfaction. *IEEE Transactions on systems, man, and cybernetics*, 21(6):1462–1477, Nov/Dec 1991.

[11] J. Denzinger. Tutorial on distributed knowledge based search. IJCAI-01, August 2001.

[12] E. H. Durfee and T. A. Montgomery. Coordination as distributed search in a hierarchical behavior space. *IEEE Transactions on systems, man, and cybernetics*, 21(6):1363–1378, Nov/Dec 1991.

[13] C. Eisenberg. Integrating a distributed and heterogeneous organisation using constraint programming. In *DCR*, 2000.

[14] B. Faltings and S. Macho-Gonzalez. Open constraint satisfaction. In *CP*, 2002.

[15] A. Felfernig, G. Friedrich, D. Jannach, and M. Zanker. Distributed configuration as distributed dynamic constraint satisfaction. In *Proc. of 14th Int. Conf. on Industrial and Engineering Applications of AI and Expert Systems*, volume 2070 of *LNAI*, pages 434–444, Budapest, June 2001. Springer.

[16] Y. Hamadi. *Traitement des problèmes de satisfaction de contraintes distribués*. PhD thesis, Université Montpellier II, Juillet 1999.

[17] M. Hannebauer. On proving properties of concurrent algorithms for distributed csps. In M.-C. Silaghi, editor, *Proceedings of the International Workshop on Distributed Constraint Satisfaction*. EPFL, 2000.

[18] K. Hirayama and M. Yokoo. An approach to over-constrained distributed constraint satisfaction problems: Distributed hierarchical constraint satisfaction. In *Proceedings of ICMAS-2000*, 2000.

[19] N. Jussien, R. Debruyne, and P. Boizumault. Maintaining arc-consistency within dynamic backtracking. In *CP'2000*, Singapore, 2000. Springer.

[20] S. Kasif and L. Delcher, Arthur. Local consistency in parallel constraint satisfaction networks. *Artificial Intelligence*, 69:307–327, 1994.

[21] J. Larrosa, P. Meseguer, and T. Schiex. Maintaining reversible dac for max-csp. *AI*, 107:149–163, 1999.

[22] J. B. Larrosa. *Algorithms and Heuristics for Total and Partial Constraint Satisfaction*. PhD thesis, IIIA,Bellaterra,Spain, 1998.

[23] M. Lemaître and G. Verfaillie. An incomplete method for solving distributed valued constraint satisfaction problems, 1997.

[24] V. R. Lesser. A retrospective view of FA/C distributed problem solving. *IEEE Transactions on systems, man, and cybernetics*, 21(6):1347–1362, Nov/Dec 1991.

[25] J. Liu and K. P. Sycara. Exploiting problem structure for distributed constraint optimization. In *ICMAS*, 1995.

[26] Q. Luo, P. Hendry, and J. Buchanan. Strategies for distributed constraint satisfaction problems. In *Proc. of the 13th International Workshop on DAI*, pages 207–221, 1994.

[27] P. J. Modi, M. Tambe, W.-M. Shen, and M. Yokoo. A general-purpose asynchronous algorithm for distributed constraint optimization. In *Distributed Constraint Reasoning, Proc. of the AAMAS'02 Workshop*, Bologna, July 2002. AAMAS.

[28] E. Monfroy and J.-H. Rety. Chaotic iteration for distributed constraint propagation. In *SAC*, pages 19–24, 1999.

[29] J. Parkes, Andrew. Exploiting solution clusters for coarse-grained distributed search. In *Proc. of IJCAI-01 DCR Workshop (CONS-2)*, pages 99–108, Seattle, August 2001.

[30] T. Petit, J. Régin, and B. C. Range-based algorithm for max-csp. In *Proceedings of CP*, pages 280–294, Ithaca, September 2002.

[31] N. Prcovic. Un algorithm distribué pour la résolution des problèmes de contraintes en domaines finis. Technical Report CERAMICS 95.44, CERAMICS, Novembre 1995.

[32] V. N. Rao and V. Kumar. On the efficiency of parallel backtracking. *IEEE*, 4(4), Apr 1993.

[33] S. Russell and P. Norvig. *Artificial Intelligence, A Modern Approach*. Prentice Hall, 2nd edition, 2002.

[34] T. Schiex, H. Fargier, and G. Verfaillie. Valued constraint satisfaction problems: hard and easy problems. In *Procs. IJCAI'95*, pages 631–637, 1995.

[35] M.-C. Silaghi. *Asynchronously Solving Distributed Problems with Privacy Requirements*. Phd thesis 2601, Swiss Federal Institute of Technology (EPFL), CH-1015 Ecublens, June 27, 2002. http://www.cs.fit.edu/~msilaghi/teza.

[36] M.-C. Silaghi, M. Calisti, and B. Faltings. On generalized english auctions. In *submitted to AAMAS-02*, page ID:393, September 2001.

[37] M.-C. Silaghi, D. Sam-Haroud, and B. Faltings. Asynchronous search with aggregations. In *Proc. of AAAI2000*, pages 917–922, Austin, August 2000.

[38] M.-C. Silaghi, D. Sam-Haroud, and B. Faltings. Fractionnement intelligent de domaine pour CSPs avec domaines ordonnés. In *Proc. of RFIA2000*, Paris, February 2000. Submitted to CP-1999.

[39] M.-C. Silaghi, D. Sam-Haroud, and B. Faltings. ABT with asynchronous reordering. In 2^{nd} *A-P Conf. on Intelligent Agent Technology*, pages 54–63, Maebashi, Japan, October 2001.

[40] M.-C. Silaghi, D. Sam-Haroud, and B. Faltings. Consistency maintenance for ABT. In *Proc. of CP'2001*, pages 271–285, Paphos,Cyprus, 2001.

[41] M.-C. Silaghi, D. Sam-Haroud, and B. Faltings. Polynomial space and complete multiply asynchronous search with abstractions. In *IJCAI-01 DCR Workshop*, pages 17–32, Seattle, August 2001.

[42] G. Solotorevsky, E. Gudes, and A. Meisels. Algorithms for solving distributed constraint satisfaction problems (DCSPs). In *AIPS96*, 1996.

[43] C. Terrioux. Cooperative search and nogood recording. In *Proc. of IJCAI-01*, pages 260–265, 2001.

[44] E. H. Turner and J. Phelps. Determining the usefulness of information from its use during problem solving. In *Proceedings of AA2000*, pages 207–208, 2000.

[45] S. Willmott and B. Faltings. The benefits of environment adaptative organizations for agent coordination and network routing problems. In *Proc. of ICMAS'00*, pages 333–340, 2000.

[46] M. Yokoo, E. H. Durfee, T. Ishida, and K. Kuwabara. The distributed constraint satisfaction problem: Formalization and algorithms. *IEEE TKDE*, 10(5):673–685, 1998.

[47] M. Yokoo and K. Hirayama. Distributed constraint satisfaction algorithm for complex local problems. In *Proceedings of 3rd ICMAS'98*, pages 372–379, 1998.

[48] W. Zhang, G. Wang, and L. Wittenburg. Distributed stochastic search for constraint satisfaction and optimization: Parallelism, phase transitions and performance. In *PAS*, 2002.