Chapter 14

Optimization

When I was a boy I was told that anybody could become President.
Now I'm beginning to believe it.
– Clarence Darrow

As described in Annex, the constraint satisfaction is only a special case of optimization. In this chapter I will show how Multiply Asynchronous Search (MAS) and Replica-based MAS can be extended and applied to optimization problems.

14.1 Distributed Valued CSPs

Constraint Satisfaction Problems (CSPs) do not model any kind of optimization requirement. An extension allowing for modeling some optimization functions is given by Valued CSPs.

Definition 14.1 (VCSP) *A Valued CSP is defined by a set, X, of variables $x_1, x_2, ..., x_m$, and a set of functions, $f_1, f_2, ..., f_i, ..., f_n$, of type $f_i : X \rightarrow \mathbb{R}$.*

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

For discrete problems with binary functions f_i, the functions can be represented by matrices with values. To model CSPs with VCSPs, infeasible tuples can be set to ∞, and feasible ones to 0. Valued CSP can be also distributed.

Definition 14.2 (DisVCSP) *A Distributed Valued CSP (DisVCSP) is defined by a set of agents $A_1, A_2, ..., A_n$, a set X of variables ,$x_1, x_2, ..., x_m$, and a set of functions $f_1, f_2, ...f_i, ..., f_n$, $f_i : X_i \rightarrow \mathbb{R}$, $X_i \subseteq X$.*

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

In practice one meets more often a distributed valued CSP than a DisCSP. Negotiations often have to deal with preferences on alternatives. Branch and Bound is a technique that can always be implemented in a constructive search technique like R-MAS.

2	5	4	7
2	6	5	4
1	5	5	7
1	2	2	∞

$\rightarrow$

1	4	4	4
1	4	4	4
1	4	4	4
1	2	2	2

$+$

1	1	0	3
1	2	1	0
0	1	1	3
0	0	0	∞

Figure 14.1: Example of constraint splitting with distributed valued CSPs.

The current proposal has to be made in such a way that the local cost is identical for all tuples of known(A_i). This can be done efficiently upon the technique of R-MAS, by allowing splitting of constraints in such a way that the all/several tuples in an abstract agent can be aggregated (see Figure 14.1).

14.2 Branch and Bound (R-MAS-BB)

The simplest way to do it is to let agents always broadcast the value of their proposal to all lower priority agents (Silaghi *et al.* 2001b). Each proposal is tagged with the cost of the agent. This can be also done by treating Cartesian products of proposals as a block with an attached cost. This alternative is called R-MAS-BB.

14.3 Branch and Bound with cost variables

Alternatively to R-MAS-BB, an elegant way to achieve branch and bound is by introducing new variables for modeling the value of the optimization function. Two alternatives are described.

14.3.1 Branch and Bound with individual cost variables (R-MAS-BB-c1)

Let us 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 and is the only variable that has to be sent to all lower priority agents.

Any solution with value C_k defines a nogood:

$$\sum_i x_{c_i} < C_k$$

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

14.3.2 Branch and Bound with shared cost variable (R-MAS-BB-c2)

Let us consider that a single new shared variable x_c, $x_c \geq 0$ is used by each agent A_i. This variable models the cost of the current proposal. Tagged with signatures for conflict resources and with the last ordering, each agent's proposal on x_c: $\langle x_c, C, h \rangle$ has the semantic $x_c \geq C$. Each agent proposes a new value C of x_c computed as the sum between the strongest received value of x_c and the local cost for each tuple of known(A_i).

Any solution with value C_k defines a nogood:

$$x_c < C_k$$

that is broadcasted to all agents. No other modification is required and again a new Branch and Bound algorithm is obtained. The last found solution is optimal. This alternative is referred to as R-MAS-BB-c2.

Alternatively, rather than broadcasting each cost modification, it can be sent only to the next lower priority agent. In this case, on the change of its strongest received cost an agent has to forward it even if the local cost for the current proposal is 0. This alternative is referred to as R-MAS-BB-c2'.

14.4 Cost of nogoods (VR-MAS)

In the previous section it can be noticed that costs are only detected when partial valuations are fully built. A better idea has been introduced in (Larrosa 1998). Larrosa explains how cost of subproblems can be computed by consistency propagation for earlier estimating bounds.

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

Definition 14.3 (SRC) *A set of references to constraints stands for a subproblem of the DisCSP, described as a set of symbols for distinct constraints.*

Due to privacy reasons, a constraint can be represented by several constraint references and several constraints of an agent can be represented by a single constraint reference.

Definition 14.4 (Valued Explicit Nogood) *A valued explicit* nogood *has the form* $\langle SRC, c, N\rangle$ *where SRC is a set of references to constraints having cost with lower bound c, given a set of assignments, N, for distinct variables.*

The (hard) explicit nogood is obtained for $c{=}\infty$.

Definition 14.5 (Valued Consistency nogood) *A valued consistency* nogood*(VCN) for a level k and a variable x has the form* $\langle SRC, c_1, c_2, V{\cup}(x{\in}l_x^k)\rangle$ *or* $\langle SRC, c_1, c_2, V \cup \neg(x{\in}s{\setminus}l_x^k)\rangle$.

V is a set of assignments. Any assignment in V must have been proposed by A_k *or its predecessors.* l_x^k *is a label,* $l_x^k{\neq}\emptyset$*. SRC is a set of references to constraints while* c_1 *and* c_2 *are low bounds of the cost of the constraints referred by SRC given V and remaining values, respectively eliminated values in* x^k*.*

Remark 14.1 *Most often* c_2 *will be* ∞*, therefore we will often use for VCNs the simplified notation* $\langle SRC, c_1, V{\cup}(x{\in}l_x^k)\rangle$ *that implies* $c_2{=}\infty$*.*

Example 14.17 *An example of a valued consistency nogood is* $\langle (C_3, C_5), 27, \infty, (\langle x_2, \{1..3\}, |1{:}0||\rangle{\cup}(x_4{\in}\{3..5\}))\rangle$*. 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_3, C_5 *is low bounded by 27 when* $x_4{\in}\{3..5\}$*, respectively low bounded by* $+\infty$ *(i.e. infeasible) otherwise.*

Definition 14.6 (Valued Conflict List (VCL)) *A valued conflict list of agent* A_i *has the form* $\langle SRC, c, N, T\rangle$ *where SRC is a set of references to constraints having cost with lower bound c, given a set of assignments, N, for distinct variables, and the set of local tuples T in the search space of* A_i*.*

The new concepts are the basis of a new family of asynchronous algorithms that extend R-MAS. We will call the new family: Valued R-MAS (VR-MAS).

Several valued nogoods can be stored for a set of aggregates and for a label. A delicate problem is the combination of valued nogoods. Two valued explicit/consistency nogoods that can be combined in R-MAS can be also combined in VR-MAS.

Definition 14.7 (valid valued nogood) *A valued nogood (valued explicit nogood, valued consistency nogood, or VCL) is valid only as long as all the aggregates involved in it are valid.*

14.4.1 Combining valued explicit nogoods

Proposition 14.1 *Any two valued explicit nogoods,* $\langle SRC_1, c_1, N_1\rangle$ *and* $\langle SRC_2, c_2, N_2\rangle$ *where any aggregates in* N_1 *and* N_2 *for the same variable are identical, can be combined into a new nogood. The obtained nogood is* $\langle SRC, c, N\rangle$ *such that* $SRC{=}SRC_1{\cup}SRC_2$*,* $c{=}max(c_1, c_2)$*, and* $N{=}N_1{\cup}N_2$*.*

The proof of this and of the following propositions follow directly from the corresponding definitions.

Proposition 14.2 *When* $SRC_1{\cap}SRC_2{=}\emptyset$*, the estimation can be tighter:* $c{=}c_1{+}c_2$*.*

This combination technique can be used in the same way as the technique for combining explicit nogoods in AAS0.

In VR-MAS, it is useful to also use the next type of inference: $\langle SRC_1, c_1, N_1\rangle{\rightarrow}\langle SRC_1, c_1, N_2\rangle$ where N_2 is obtained from N_1 by adding additional aggregates that do not invalidate aggregates in N_1. When aggregates are added for the same variable, only the strongest one is retained.

14.4.2 Combining valued conflict lists

Proposition 14.3 *Any two valued conflict lists, $\langle SRC_1, c_1, N_1, T_1\rangle$ and $\langle SRC_2, c_2, N_2, T_2\rangle$ where any aggregates in N_1 and N_2 for the same variable are identical, can be combined into a new VCL. The obtained VCL is $\langle SRC, c, N, T\rangle$ such that $SRC{=}SRC_1{\cup}SRC_2$, $c{=}min(c_1, c_2)$, $N{=}N_1{\cup}N_2$, and $T{=}T_1{\cup}T_2$.*

This combination technique can be used in the same way as the technique for computing new explicit nogoods in AAS, or for building the conflict list in AAS0.

In VR-MAS, it is useful to also use the next types of inferences:

- Valued Explicit Nogood → Valued Conflict List.

  For example, the following inference can be made by an agent A_i:

  $$\langle SRC, c, N{\cup}V\rangle \rightarrow \langle SRC, c{+}cost_i(T), N, T\rangle$$

  where N are aggregates that are not built by A_i while V are aggregates built by A_i. T is the set of tuples of A_i that are covered by $N{\cup}V$. $cost_i(T)$ is the minimal cost for A_i of a tuple in T.

- Valued Conflict List → Valued Explicit Nogood

  For example, the following inference can be made by an agent A_i:

  $$\langle SRC, c, N, SS(A_i)\rangle \rightarrow \langle SRC \cup \{CR(A_i)\}, c, N\rangle$$

  where N are aggregates that are not built by A_i, $SS(A_i)$ is the search space of A_i (see Definition 9.15), and $CR(A_i)$ is a reference to the constraints of A_i.

Incoming valid Valued Explicit Nogood can be stored or translated and merged in a Valued Conflict List. When no new proposal is possible, a Valued Conflict List covering the whole search space is computed and is the transfered back in a new Valued Explicit Nogood that is sent to an agent as in MAS.

14.4.3 Combining valued consistency nogoods

Proposition 14.4 *Any two valued 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 valued 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.*

Remark 14.2 *A stronger VCN can be computed in Proposition 14.4 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 14.4.1 *Any two valued 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 valued 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 14.5 *When $SRC_1{\cap}SRC_2{=}\emptyset$ in Proposition 14.4, the estimation can be tighter: $c{=}c_1{+}c_2$, $c'{=}min(c'_1{+}c_2, c'_2{+}c_1, c'_1{+}c'_2)$.*

14.4.4 Combining valued consistency and explicit nogoods

Proposition 14.6 *Any valued consistency nogood, $\langle SRC_1, c_1, N_1 \cup x \in l_1 \rangle$, can be combined with a valued explicit nogood $\langle SRC_2, c_2, N_2 \rangle$ where any aggregates in N_1 and N_2 for the same variable do not invalidate each other, can be combined into a new valued consistency nogood. The obtained nogood is $\langle SRC, c, N \cup x \in l \rangle$ such that $SRC = SRC_1 \cup SRC_2$, $c = max(c_1, c_2)$, and $N = N_1 \cup N_2$, N retaining only the strongest among two aggregates for the same variable. l is obtained from l_1 by intersection with any assignment for x found in N_2.*

Proposition 14.7 *When $SRC_1 \cap SRC_2 = \emptyset$, the estimation can be tighter: $c = c_1 + c_2$.*

This combination technique can also be used in the same way as the technique for combining backtracking and consistency nogoods in DMAC.

14.4.5 Tight combination of valued nogoods (VR-MAS$_1$)

In order to benefit from tight combinations, the solution is to maintain several valued nogoods for distinct subsets of references to constraints. A notable polynomial space heuristic is to always keep separate valued nogoods for each reference to a constraint. The space requirement only multiplies with a factor equal to the total number of references to constraints.

14.4.6 Extended Branch and Bound algorithm (VR-MAS)

Valued Replica-based MAS uses the previously mentioned form for valued nogoods instead of classic R-MAS-BB-c2 nogoods. The cost C_k of any solution is broadcast under the form of a nogood

$$x_c < C_k,$$

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

VR-MAS starts with all agents enforcing a constraint $x_c < \infty$.

A valued explicit nogood can be obtained via a VCL, as shown above.

An agent has to abandon any of its proposals involved in a conflict with a constraint $x_c < C_k$. When no proposal is available, the agent generates a valued explicit nogood for elements of its *agent view* by combining the valued explicit nogoods for its proposals, augmented with its own cost for each proposal.

A valued consistency nogood can be generated by proving that a certain value 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 $x_c < C_k$.

14.4.7 Dynamic Optimization (DVR-MAS)

In (Silaghi *et al.* 2001m) (see also Section 17.9) we explained that there are two key elements that are required for improving efficiency for optimization with large problems:

- Limiting commitment (Section 17.9.1). This consists in abandoning a branch if it is not promising.
- Using acceptable value ordering heuristics (Remark 17.2).

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 (Modi *et al.* 2002) 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 14.3 *To avoid that two much work is lost too often by discarding nogoods due to frequently abandoning commitments (phenomena that I have 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 14.4 *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 (see Annex A).*

An optimistic estimate of the cost of a branch can be obtained in VR-MAS as follows.

Remark 14.5 *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 cost c of the most expensive VCL, $\langle SRC, c, N, B\rangle$, that can be inferred.*

The following Lemma can be useful for estimating costs:

Lemma 14.1 *$\forall SRC, c \in \mathbb{R}, N, B$, where SRC is a set of references to constraints, c is their lower bound cost given a view N and a set of tuples B, if $\langle SRC, c, N, B\rangle$ is a VCL, then one can infer $\langle SRC, c, N, B'\rangle$, $\forall B' \subset B$.*

There is a straightforward way to implement a value ordering heuristic in VR-MAS (also used in (Modi *et al.* 2002)).

Remark 14.6 *When the commitment on a branch is abandoned, the value ordering heuristic to be used in agreement with A^* is to propose the branch (aggregate-set) that has the lowest estimated cost, within the current level of abstraction of the agent/replica.*

The obtained algorithm is called DVR-MAS. Distributed optimization is one of the most hot topics of the moment, and many colleagues perform research with optimization for DisVCSPs. A related work is the one I performed with Flavio Ferri in 2000 during his semester project on VCSP optimization with A^*.

14.4.8 Canonic Notation

The notation introduced in previous chapters for distinguishing versions of RMAS can be extended to take into account the features of the optimization.

The basic alternatives in optimization are:

z_0 no treatment of cost (CSP — no optimization)

z_1 individual variables for costs (as R-MAS-BB-c1)

z_2 a single variable for cost (as R-MAS-BB-c2)

z_3 a single variable without broadcast (as R-MAS-BB-c2')

The notation I propose is described by the regular expression:
$[[\mathrm{D}]\mathrm{V}]\mathrm{R}\text{-}]\mathrm{MAS}_{(\mp\mp\mp\mp)} - (p^{[']}_{(1|2)}|r^{[']})[(*|l|s|d)[e]] - [(g|v|h)^{[']|''|'''][k]}[A|C][o|b|i|n]] - [z_0|z_1|z_2|z_3]$.
The meaning of each element of the notation is the same as in section 13.6.

14.5 Summary

In this chapter we have seen that the techniques presented in previous chapters of this thesis can be extended easily to distributed optimization problems. The efficiency of the algorithms I propose here remain to be evaluated.