Chapter 1

Introduction

Any beginning has an end.
Folklore

THIS thesis addresses Asynchronous Search Techniques for Distributed Constraint Satisfaction. Many techniques employed in asynchronous search are inspired from centralized algorithms. Those issues are simpler and can be easier understood when they are presented in centralized settings. This is why in its first part, this thesis starts by building such a background.

Most of the time, parallelism/distribution is employed uniquely to speed up initially centralized problems. Instead, here we are confronted with distribution from a completely different point of view: *We discuss approaches to problems which due to requirements of privacy cannot be centralized.* Robustness, security and openness requirements can also motivate recourse to distributed protocols. The distributed satisfaction problems are defined by sets of constraints that are not all known to one and the same reasoning process.

size\technology	SIPP	DIMM
128 MB	X	
256 GB		X

Seller (Agent 1)

if(size $>$ 128 GB) $a + b == 14$
if(technology is DIMM) $a + b < 20$

Buyer (Agent 2)

Figure 1.1: The RAM seller&buyer problem is a distributed problem consisting of two agents. The agents have access to the constraints depicted in the corresponding ovals.

if(size $>$ 128 GB) $a + b == 14$
if(technology is DIMM) $a + b < 20$

size\technology	SIPP	DIMM
128 MB	X	
256 GB		X

Figure 1.2: (Centralized) version of the problem in Figure 1.1

Our way of differentiating between centralized, parallel, and distributed problems is exemplified in Figures 1.1, 1.2, and 1.3. Figure 1.1 shows a problem that models a seller (Agent 1) and a buyer (Agent 2). The seller has a list of available RAM chips manufactured by different producers. The buyer wants to integrate some chips into his computer prototype and reasons about the technical features a and b. The corresponding constraints of each of the involved parts are shown in the

if(size $>$ 128 GB) $a + b == 14$
if(technology is DIMM) $a + b < 20$

size\technology	SIPP	DIMM
128 MB	X	
256 GB		X

Process 1 (Agent 1)

if(size $>$ 128 GB) $a + b == 14$
if(technology is DIMM) $a + b < 20$

size\technology	SIPP	DIMM
128 MB	X	
256 GB		X

Process 2 (Agent 2)

Figure 1.3: Parallel solving of the problem in Figure 1.2

corresponding ovals of Figure 1.1. This thesis concentrates on the techniques for solving such problems without requiring the involved parts to completely exchange their sets of constraints.

Many approaches for solving problems similar to the RAM buyer&seller problem need to gather somewhere the constraints of all involved agents and then solve the obtained centralized problem (Figure 1.2):

- *Centralized Solution (seller)* The buyer gives his constraints to the seller and the seller proposes a solution. This may not be acceptable for the buyer if his technology is not yet public.

- *Centralized Solution (buyer)* The seller gives his complete offer to the buyer. The buyer chooses the configuration that suits him best. This may not be acceptable for the seller if there is not much competition on the market. The seller would then prefer to push the customer towards buying products that are expensive, less required, and that he owns in large quantities.

- *Centralized Solution (Truth incentive)* Both involved parties reveal their constraints based on a truth incentive mechanism (this means a mechanism where the agents cannot gain more than by telling the whole truth). Both parties can solve the problem separately in a way that allows them to reach the same solution. Both secrets are lost.

- *Centralized Solution (Trusted party)* Both involved parties send their constraints to a trusted neutral solver, via secure channels. The trusted party finds a set of solutions which are returned simultaneously to both seller and buyer. This approach improves on the previous ones, whenever a trusted party can be defined.

  However, people don't trust secure channels over the Internet (many people are afraid to send their Credit Card numbers). Servers (even secure and very trusted ones) are often prey to all kind of viruses and attacks.

  For hard problems, heuristics have to be used by the trusted solver leading to suboptimal decisions. Consequently some involved party undergoes a disfavor. A loss due to the uncontrollable sub-optimality of the trusted party is often less acceptable than a loss due to the inefficient negotiation of the loser.

- *Distributed trusted party with private constraints* The involved parties encode their constraints and distribute them to a set of independent agents defining a trusted party. The trusted party uses a secure protocol equivalent to some centralized algorithm. The efficiency advantage of the trusted party solution is lost. However, the drawback coming from the lack of confidence in a single trusted party is replaced with a smaller drawback: confidence in a majority of a set of trusted parties.

When any of the previously mentioned solutions is chosen, the resolution can be done using a centralized algorithm. Sometimes, for efficiency reasons, algorithms for centralized problems resort to parallelism. Parallelism can be obtained by offering several processes access to the centralized problem (see Figure 1.3). Efficient parallelism can often be obtained by distributing to each of these processes access only to loosely coupled subparts of the initial problem.

Usually the initial distribution of a problem[1] is not the same with its most efficient redistribution for a suitable distributed algorithm (Amir & McIlraith 2001; Monfroy & Rety 1999; Hirayama *et al.* 2000). However, when such algorithms can solve the initial distribution of the problems, they can be used to ameliorate the aforementioned insufficiencies of the centralization based methods.

- *Distributed solving with private constraints* The involved parties negotiate by exchanging processed constraints until a solution or failure is reached. The drawback of this approach is that the negotiation may let one partner to partially or even completely reconstruct another's problem. The agents negotiate the amount of information that they give away.

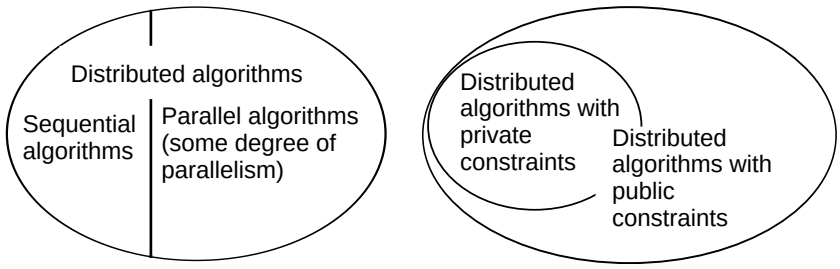

Figure 1.4: a)Distributed algorithms are either sequential or have some degree of parallelism. b)The distributed algorithms with private constraints can be used for solving problems with public constraints.

Distributed algorithms may offer more or less parallelism (Figure 1.4a). Some distributed algorithms are completely sequential (e.g. Synchronous Backtracking, first presented in (Yokoo *et al.* 1992)). Distributed algorithms (even inefficient ones) still can make sense for *distributed solving with private constraints* or for distributed problems whose constraints cannot be centralized due to their nature: e.g. the output of a remote device, highly important secret constraints, dynamic constraints.

This thesis is a detailed introduction to state of the art distributed algorithms with private constraints. All distributed algorithms with private constraints can be used for solving problems with public constraints (Figure 1.4b) and for solving centralized problems. It is generally clear that for most centralizable problems there exist distributed algorithms that deal with redistributions of the initial private constraints more efficiently than any distributed algorithm could deal with the initial distribution of the problems.

The privacy usually refers to revelations concerning values of constraint tuples. Here we will refer to tuples of the total constraint of an agent. When the participant, A, in a negotiation has several internal constraints (constraints with different sources or semantics), the total constraint of A is the constraint obtained by the composition of all its internal constraints. Each constraint tuple can be associated with a privacy value. Depending on problem, the privacy values of constraint tuples can be very different from one tuple to another and most often it is not known. The most fair comparison between negotiation protocols consists in considering each private tuple revelation as a separate cost. Some protocols change the nature of information exchange such that most revelations correspond to a reduced privacy loss (Section 13).

Besides certain degrees of privacy and parallelism, another reason for using some distributed solvers relies in some additional robustness that is exhibited especially by asynchronous algorithms. Asynchronous distributed algorithms are robust to timing variations and can prove unsatisfiability even in the presence of crashed agents (Silaghi *et al.* 2001i). The replicated servers are deeply studied due to their robustness to openness allowing for servers to leave, and the distributed solvers are expected to offer at least a few similar properties.

The last but not least motivation for studying and developing asynchronous protocols is their generality. It can be easily shown that asynchronous protocols are more general than asynchronous

[1]This is given in Figure 1.1 for the RAM seller&buyer problem.

algorithms, which are in their turn more general than synchronous algorithms. All distributed algorithms are more general than the equivalent centralized algorithms. As with quantum computing one checks many combinations at once, proving properties for asynchronous protocols proves properties for many algorithms at once.

Some other definitions to distribution and parallelism are reviewed in Chapter 3.

1.1 Guide to read this thesis

This thesis was mainly written during May-June 2001, and March-April 2002, but a few small changes and updates have been done between these more intense periods. When the Annex A was written in April 2001, I described optimization as a future trend in distributed constraints satisfaction. The fact that now, one year later, it should rather be described as a state of the art, is one of the hints of the dynamism of the described topics. Taking time to update those elements, would only introduce delays leading to new anachronisms. You should rather see this thesis as a sequence of snapshots taken between April 2001 and April 2002.

The thesis has been organized in four parts. In Annex A and in the first part we position the asynchronous Distributed Constraint Satisfaction techniques in the world of algorithms. This is important not only for a better understanding of the state of the art but also for grasping the current trends in Distributed Constraint Satisfaction. Annex A reviews the major types of general approaches to solving numeric problems. The reader that has a global view of numeric techniques, or that is not interested in understanding the position of Distributed Constraint Satisfaction approaches in the panel of numerical techniques, can start by reading the Chapter 2. In the following chapters we detail relevant topics developed first for centralized sequential Constraint Satisfaction. The reader that is familiar with Constraint Satisfaction can jump directly to Chapter 3. Any reader that is already familiar to Distributed Constraint Satisfaction can start directly with Chapter 7, found in the second part.

Part two describes several major contributions to asynchronous search protocols for DisCSPs. It introduces progressively asynchronous protocols. It presents recent algorithms with new sources of parallelism and that are also generalizations of existing ones. They are the main contribution of this thesis. The third part is intended to analyze some properties of distributed search algorithms. It also details advanced issues for known applications and presents some technical details related to distributed systems. Annex D discusses some marginal issues on the aggregation in pure centralized settings.

Handling asynchronous Distributed CSP Algorithms requires a good understanding of the mechanisms of complete search. These mechanisms are highlighted by Dynamic Backtracking, presented in Chapter 2 together with some extensions. Another important element is the local consistency that is similarly introduced in the same chapter. A deeper understanding of the notion of locality can be found in Annex A.

An introduction and the state of the art in asynchronous complete search algorithms for distributed constraint satisfaction problems is given in the following two chapters. Here are also presented the elements describing the settings of our evaluations.

Chapter 5 describes some centralized techniques that can be used by agents for their local computations in order to fully exploit the power of new distributed protocols.

The most important technique for constraint satisfaction is the *local consistency*, a technique that can bring exponential improvements with polynomial cost. *Consistency maintenance* is a technique that increases exploitation of *local consistency* by some dynamic detection of opportunities. Chapter 7 describes a generalization of consistency maintenance that brings new opportunities for exploiting local consistency. Moreover, the generalization allows the usage of consistency maintenance in asynchronous algorithms.

Chapter 8 introduces a theoretical framework for two additional major contributions, described in Chapters 9 and 10: AAS and ABTR. AAS extends existing protocols with a proper treatment of intervals/sets in search. This is an essential step towards proper treatment of numeric constraints, but also a source of efficiency. ABTR is the first technique to introduce dynamic reordering in

polynomial space algorithms. The power of modeling with abstract (virtual) agents is illustrated by ShABT that introduces the use of consensus/voting mechanisms in choosing instantiations.

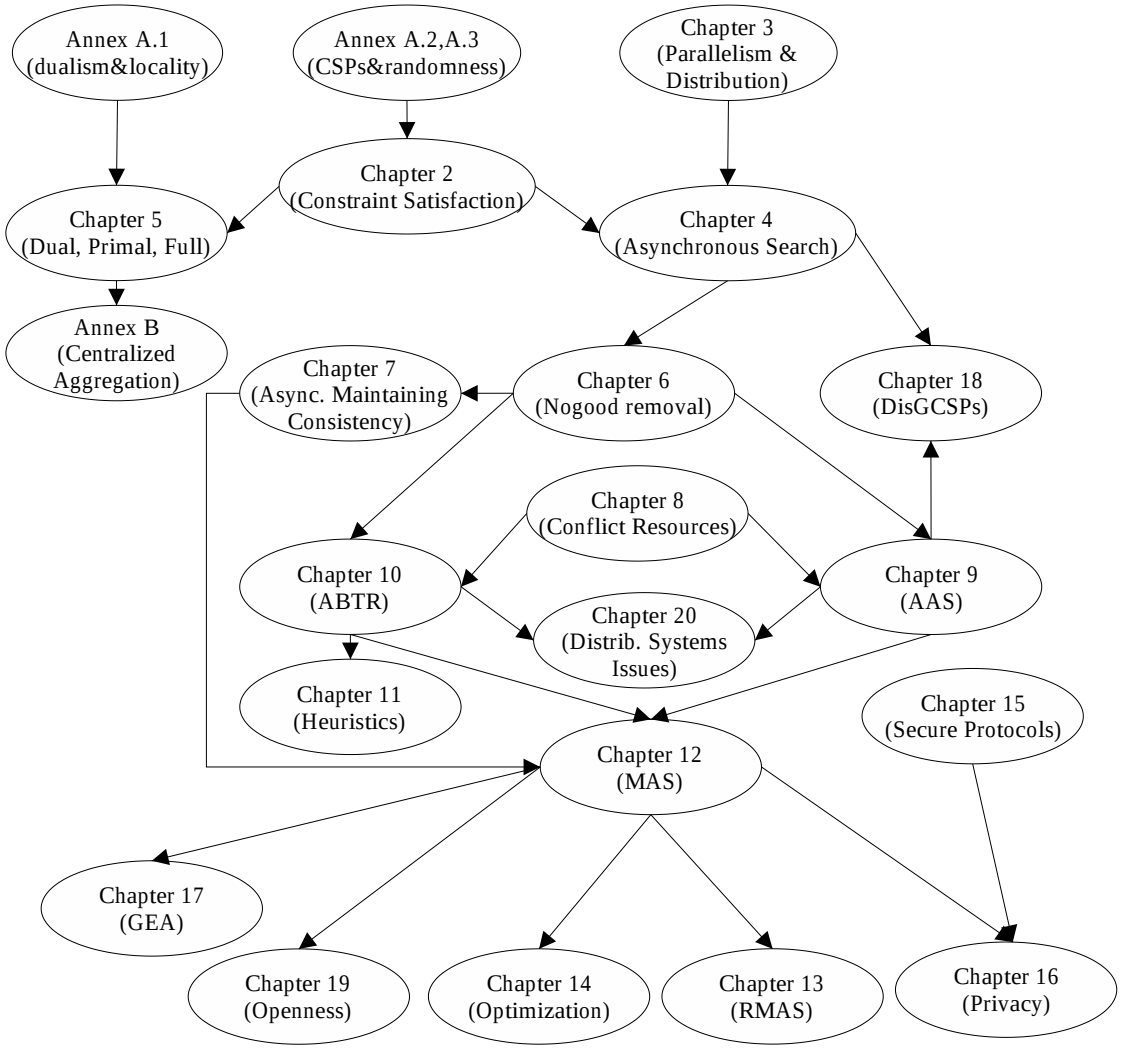

Figure 1.5: Graph of dependencies between chapters.

The following chapter shows some heuristics that can be used for reordering in ABTR. Due to the small size of the tested problems, the efficiency improvements obtained so far are not convincing. However, some convincing applications of reordering that I succeeded to find are described in later parts (e.g. SAS). The rest of the second part describes modeling techniques that we recommend and proves the simplicity of combining the described techniques (Multiply Asynchronous Search).

Chapter 14 shows how all the techniques contributed up to that chapter can be used for optimization.

The Chapter 16 introduces a framework allowing for a theoretical comparison of existing asynchronous search algorithms with respect to privacy. A comparison with some existing and contributed cryptographic protocols reveal trade-offs defining the applicability of asynchronous search techniques (Chapter 15).

The last part focuses on extensions required for two applications: Generalized English Auctions and Distributed Configuration. A relaxation technique for negotiation problems is proposed in Section 17.10.

The Chapter 19 describes few practical elements in implementations of our technique, while

the last chapter describes techniques and contributions related to generic research in Distributed Systems.

Figure 1.5 gives a detail description of the dependencies between different chapters. Please note that the dependency relation chapters is the transitive closure of the illustrated one. Nevertheless, some dependencies are due only to small parts of some chapters (e.g. some definitions or details).

1.2 Notations and Conventions

... and whatever Man called each living soul, that was its name.
Moses, Genesis, 2:19

The abbreviations used in this thesis are summarized in the next table:

AAS	Asynchronous Aggregation Search
AASBB	Asynchronous Aggregation Search with Branch and Bound
AASR	Asynchronous Aggregation Search with Reordering
ABT	Asynchronous Backtracking
ABTR	ABT with Asynchronous Reordering
AC	arc-consistency
ADMC	Asynchronous Dichotomous Maintaining Consistency
AWC	Asynchronous Weak-Commitment
BC	bound-consistency
CDB	CPR-based Dynamic Backtracking
CPR	Cartesian Product Representation
CSP	Constraint Satisfaction Problem
DDB	Dependency Directed Backtracking
DisCSP	Distributed CSP
DisDyCSP	Distributed Dynamic CSP
DyDisCSP	Dynamic DisCSP
DC	Distributed Consistency
DFC	Distributed Forward Checking
DIBT	Distributed Backtracking
DRS	Dynamic replica spawning
DMAC	Distributed Maintaining Asynchronous Consistency
DVR-MAS	Dynamic Valued Replica-based MAS
FBT	Full Backtracking
GOB	POB-like version of the Generalized Partial Order Dynamic Backtracking
GPB	General Partial Order Dynamic Backtracking
ITM	Interactive Turing Machine
LO	Locality of the guarantees
LP	Locality in the problem
LS	Locality in physical space
MAC	Maintaining Arc Consistency
MAS	Multiply Asynchronous Search
MAS-PrivRelax	Multiply Asynchronous Search with Relaxation of Private Constraints
MHDC	Maintaining Hierarchical Distributed Consistency

NCSP	Numeric CSP
NVCSP	Negotiation Valued CSP
PDB	Partial Order Dynamic Backtracking
POB	Partial Order Backtracking
RDisCSP	Replica-based DisCSP
R-MAS	Replica-based MAS
R-MAS-BB	Replica-based MAS with Branch and Bound
R-MAS-BB-c*	Replica-based MAS with Branch and Bound and cost variables
SAS	Asynchronous Secure Search with Branch and Bound
SASBB	Asynchronous Secure Search
SBMP	Secure backtracking with Merritt's protocol
SRC	Set of References to the Constraints
SRCCS	Set of References to the Constraints Completely Satisfied
SSS	Synchronous Secure Search
VCSP	Valued CSP
VR-MAS	Valued Replica-based MAS

The next notations are used frequently:

A, A_i	agent
CSP(A)	the CSP known by A
ECSP(A)	the CSP enforced by A
vars(A)	the variables in CSP(A)
evars(A)	the variables in ECSP(A)
$A^i(o)$	the agent with position i in the sequence of agents o
x, x_i	variable
$x^i(o)$	in ABTR, the variable with position i in given the sequence of agents o
$\langle x, s, h\rangle$	a tuple: variable, set of values, signature
$\langle x, v\rangle$	a tuple: variable, value
$\langle x, v, t\rangle$	a tuple: variable, value, tag
$\mathcal{O}$, $\langle o, h\rangle$	a tuple: sequence of agent names, signature
$\mathcal{O}_i(A_j)$	order known by A_j for A^i
V, V_i, V_i^j	an aggregate-set, a view
$\|a{:}b\|$	a pair in signature
$\|\|$	an empty signature
h, $\|i{:}k_i\|j{:}k_j\|...$	a signature
o, o_i	a sequence of agents, ordering
$\mathbb{R}$	the set of reals
$\mathbb{N}$	the set of integer numbers
$\mathbb{N}_+$	the set of non-negative integer numbers
$\mathbb{F}$	the set of floating point numbers
$\mathbb{I}$	the set of intervals over reals
$\mathcal{F}$	the set of floating point minimal intervals
$f^{(-1)}$	the inverse of the function f
tuple.index	the indexth element of the tuple
tuple.variable	the element of *tuple* (describing a valuation) that corresponds to *variable*
tuple$_{\|x}$	the element of the *tuple*, that corresponds to the variable x

Part I

Background and Preliminaries

This part presents related and necessary background in centralized problem solving. After reading this part (and eventually the Annexes), you are expected to know:

- The main problem definitions, frameworks, and related concepts in centralized settings.
- The main types of algorithms for solving numeric problems, and especially constraint satisfaction problems.
- The main research direction in centralized problems, directions that port to distributed settings.

The contribution of the thesis described in this part consists mainly in a bound consistency algorithm (BC1999) (Chapter 2), some numeric search algorithms (UCA6) and a framework presented in Chapter 5 (FBT).