

# **FOUNDATIONS OF CONSTRAINT SATISFACTION**

**Edward Tsang**  
Department of Computer Science  
University of Essex  
Colchester  
Essex, UK

Copyright 1996 by Edward Tsang

All rights reserved. No part of this book may be reproduced in any form by photostat, microfilm, or any other means, without written permission from the author.

Copyright 1993-95 by  
Academic Press Limited

This book was first published by  
Academic Press Limited in 1993 in  
UK: 24-28 Oval Road, London NW1 7DX  
USA: San Diego, CA 92101  
ISBN 0-12-701610-4

*To Lorna*



# Preface

Many problems can be formulated as Constraint Satisfaction Problems (CSPs), although researchers who are untrained in this field sometimes fail to recognize them, and consequently, fail to make use of specialized techniques for solving them. In recent years, constraint satisfaction has come to be seen as the core problem in many applications, for example temporal reasoning, resource allocation, scheduling. Its role in logic programming has also been recognized. The importance of constraint satisfaction is reflected by the abundance of publications made at recent conferences such as IJCAI-89, AAAI-90, ECAI-92 and AAAI-92. A special volume of *Artificial Intelligence* was also dedicated to constraint reasoning in 1992 (Vol 58, Nos 1-3).

The scattering and lack of organization of material in the field of constraint satisfaction, and the diversity of terminologies being used in different parts of the literature, make this important topic more difficult to study than is necessary. One of the objectives of this book is to consolidate the results of CSP research so far, and to enable newcomers to the field to study this problem more easily. The aim here is to organize and explain existing work in CSP, and to provide pointers to frontier research in this field. This book is mainly about algorithms for solving CSPs.

The volume can be used as a reference by artificial intelligence researchers, or as a textbook by students on advanced artificial intelligence courses. It should also help knowledge engineers apply existing techniques to solve CSPs or problems which embed CSPs. Most algorithms described in this book have been explained in pseudo code, and sometimes illustrated with Prolog codes (to illustrate how the algorithms could be implemented). Prolog has been chosen because, compared with other languages, one can show the logic of the algorithms more clearly. I have tried as much as possible to stick to pure Prolog here, and avoid using non-logical constructs such as assert and retract. The Edinburgh syntax has been adopted.

CSP is a growing research area, thus it has been hard to decide what material to include in this book. I have decided to include work which I believe to be either fundamental or promising. Judgement has to be made, and it is inevitably subjective. It is quite possible that important work, especially current research which I have not been able to fully evaluate, have been mentioned too briefly, or completely missed out.

An attempt has been made to make this book self-contained so that readers should need to refer to as few other sources as possible. However, material which is too lengthy to explain here, but which has been well documented elsewhere, has been left out.

Formal logic (mainly first order predicate calculus) is used in definitions to avoid ambiguity. However, doing so leaves less room for error, therefore errors are inevitable. For them, I take full responsibility.

Edward Tsang  
University of Essex, UK

# Acknowledgements

I am grateful to Jim Doran for bringing me into the topic of constraint satisfaction. Sam Steel suggested me to write this book. Ray Turner and Nadim Obeid advised me on a number of issues. Hans Guesgen and Joachim Hertzberg generously gave me a copy of their book on this topic and discussed their work with me. Patrick Prosser read an earlier draft of this book in detail, and gave me invaluable feedback. Barbara Smith, Barry Crabtree and Andrew Davenport all spared their precious time to read an earlier draft of this book. I would like to thank them all. My special thanks goes to Alvin Kwan, who has read earlier versions of this book and had lengthy discussions with me on many issues. The Department of Computer Science, University of Essex, has provided me with a harmonious environment and a great deal of support. Feedback from students who took my course on constraint satisfaction has been useful. Andrew Carrick, Kate Brewin and Nigel Eyre made the publication of this book a relatively smooth exercise. Most importantly, I would like to thank my wife Lorna. Without her support this book could never have been completed.





# Table of contents

<b>Preface</b>	<b>v</b>
<b>Acknowledgements</b>	<b>vii</b>
<b>Table of contents</b>	<b>ix</b>
<b>Figures</b>	<b>xv</b>
<b>Notations and abbreviations</b>	<b>xix</b>
<b>Chapter 1 Introduction</b>	<b>1</b>
1.1 What is a constraint satisfaction problem?	1
1.1.1 Example 1 —The N-queens problem	1
1.1.2 Example 2 — The car sequencing problem	3
1.2 Formal Definition of the CSP	5
1.2.1 Definitions of domain and labels	5
1.2.2 Definitions of constraints	7
1.2.3 Definitions of satisfiability	8
1.2.4 Formal definition of constraint satisfaction problems	9
1.2.5 Task in a CSP	10
1.2.6 Remarks on the definition of CSPs	10
1.3 Constraint Representation and Binary CSPs	10
1.4 Graph-related Concepts	12
1.5 Examples and Applications of CSPs	17
1.5.1 The N-queens problem	17
1.5.2 The graph colouring problem	19
1.5.3 The scene labelling problem	21
1.5.4 Temporal reasoning	24
1.5.5 Resource allocation in AI planning and scheduling	25
1.5.6 Graph matching	26
1.5.7 Other applications	26
1.6 Constraint Programming	27
1.7 Structure Of Subsequent Chapters	28
1.8 Bibliographical Remarks	29
<b>Chapter 2 CSP solving — An overview</b>	<b>31</b>
2.1 Introduction	31
2.1.1 Soundness and completeness of algorithms	31
2.1.2 Domain specific vs. general methods	32
2.2 Problem Reduction	32

2.2.1	Equivalence	32
2.2.2	Reduction of a problem	33
2.2.3	Minimal problems	34
2.3	Searching For Solution Tuples	35
2.3.1	Simple backtracking	36
2.3.2	Search space of CSPs	38
2.3.3	General characteristics of CSP's search space	40
2.3.4	Combining problem reduction and search	41
2.3.5	Choice points in searching	42
2.3.6	Backtrack-free search	43
2.4	Solution Synthesis	44
2.5	Characteristics of Individual CSPs	46
2.5.1	Number of solutions required	46
2.5.2	Problem size	47
2.5.3	Types of variables and constraints	47
2.5.4	Structure of the constraint graph in binary- constraint-problems	47
2.5.5	Tightness of a problem	48
2.5.6	Quality of solutions	49
2.5.7	Partial solutions	50
2.6	Summary	51
2.7	Bibliographical Remarks	52
<b>Chapter 3</b>	<b>Fundamental concepts in the CSP</b>	<b>53</b>
3.1	Introduction	53
3.2	Concepts Concerning Satisfiability and Consistency	54
3.2.1	Definition of satisfiability	54
3.2.2	Definition of k-consistency	55
3.2.3	Definition of node- and arc-consistency	57
3.2.4	Definition of path-consistency	59
3.2.5	Refinement of PC	60
3.2.6	Directional arc- and path-consistency	63
3.3	Relating Consistency to Satisfiability	64
3.4	(i, j)-consistency	68
3.5	Redundancy of Constraints	69
3.6	More Graph-related Concepts	70
3.7	Discussion and Summary	76
3.8	Bibliographical Remarks	76

<b>Chapter 4 Problem reduction</b>	<b>79</b>
4.1 Introduction	79
4.2 Node and Arc-consistency Achieving Algorithms	80
4.2.1 Achieving NC	80
4.2.2 A naive algorithm for achieving AC	81
4.2.3 Improved AC achievement algorithms	83
4.2.4 AC-4, an optimal algorithm for achieving AC	84
4.2.5 Achieving DAC	88
4.3 Path-consistency Achievement Algorithms	90
4.3.1 Relations composition	91
4.3.2 PC-1, a naive PC Algorithm	92
4.3.3 PC-2, an improvement over PC-1	93
4.3.4 Further improvement of PC achievement algorithms	95
4.3.5 GAC4: problem reduction for general CSPs	99
4.3.6 Achieving DPC	99
4.4 Post-conditions of PC Algorithms	101
4.5 Algorithm for Achieving k-consistency	102
4.6 Adaptive-consistency	105
4.7 Parallel/Distributed Consistency Achievement	110
4.7.1 A connectionist approach to AC achievement	110
4.7.2 Extended parallel arc-consistency	112
4.7.3 Intractability of parallel consistency	115
4.8 Summary	115
4.9 Bibliographical Remarks	117
<b>Chapter 5 Basic search strategies for solving CSPs</b>	<b>119</b>
5.1 Introduction	119
5.2 General Search Strategies	120
5.2.1 Chronological backtracking	120
5.2.2 Iterative broadening	121
5.3 Lookahead Strategies	124
5.3.1 Forward Checking	124
5.3.2 The Directional AC-Lookahead algorithm	130
5.3.3 The AC-Lookahead algorithm	133
5.3.4 Remarks on lookahead algorithms	136
5.4 Gather-information-while-searching Strategies	136
5.4.1 Dependency directed backtracking	137
5.4.2 Learning nogood compound labels algorithms	143

5.4.3 Backchecking and Backmarking	147
5.5 Hybrid Algorithms and Truth Maintenance	151
5.6 Comparison of Algorithms	152
5.7 Summary	155
5.8 Bibliographical Remarks	155
<b>Chapter 6 Search orders in CSPs</b>	<b>157</b>
6.1 Introduction	157
6.2 Ordering of Variables in Searching	157
6.2.1 The Minimal Width Ordering Heuristic	158
6.2.2 The Minimal Bandwidth Ordering Heuristic	166
6.2.3 The Fail First Principle	178
6.2.4 The maximum cardinality ordering	179
6.2.5 Finding the next variable to label	180
6.3 Ordering of Values in Searching	184
6.3.1 Rationale behind values ordering	184
6.3.2 The min-conflict heuristic and informed backtrack	184
6.3.3 Implementation of Informed-Backtrack	187
6.4 Ordering of Inferences in Searching	187
6.5 Summary	187
6.6 Bibliographical Remarks	188
<b>Chapter 7 Exploitation of problem-specific features</b>	<b>189</b>
7.1 Introduction	189
7.2 Problem Decomposition	190
7.3 Recognition and Searching in k-trees	192
7.3.1 “Easy problems”: CSPs which constraint graphs are trees	192
7.3.2 Searching in problems which constraint graphs are k-trees	194
7.4 Problem Reduction by Removing Redundant Constraints	200
7.5 Cycle-cutsets, Stable Sets and Pseudo_Tree_Search	201
7.5.1 The cycle-cutset method	201
7.5.2 Stable sets	207
7.5.3 Pseudo-tree search	209
7.6 The Tree-clustering Method	212
7.6.1 Generation of dual problems	212
7.6.2 Addition and removal of redundant constraints	214

7.6.3 Overview of the tree-clustering method	216
7.6.4 Generating chordal primal graphs	222
7.6.5 Finding maximum cliques	224
7.6.6 Forming join-trees	231
7.6.7 The tree-clustering procedure	234
7.7 j-width and Backtrack-bounded Search	235
7.7.1 Definition of j-width	235
7.7.2 Achieving backtrack-bounded search	239
7.8 CSPs with Binary Numerical Constraints	240
7.8.1 Motivation	241
7.8.2 The AnalyseLongestPaths algorithm	243
7.9 Summary	245
7.10 Bibliographical Remarks	250
<b>Chapter 8 Stochastic search methods for CSPs</b>	<b>253</b>
8.1 Introduction	253
8.2 Hill-climbing	254
8.2.1 General hill-climbing algorithms	254
8.2.2 The heuristic repair method	256
8.2.3 A gradient-based conflict minimization hill-climbing heuristic	258
8.3 Connectionist Approach	261
8.3.1 Overview of problem solving using connectionist approaches	261
8.3.2 GENET, a connectionist approach to the CSP	261
8.3.3 Completeness of GENET	266
8.3.4 Performance of GENET	267
8.4 Summary	268
8.5 Bibliographical Remarks	269
<b>Chapter 9 Solution synthesis</b>	<b>271</b>
9.1 Introduction	271
9.2 Freuder's Solution Synthesis Algorithm	272
9.2.1 Constraints propagation in Freuder's algorithm	273
9.2.2 Algorithm Synthesis	274
9.2.3 Example of running Freuder's Algorithm	276
9.2.4 Implementation of Freuder's synthesis algorithm	279
9.3 Seidel's Invasion Algorithm	280

9.3.1	Definitions and Data Structure	280
9.3.2	The invasion algorithm	282
9.3.3	Complexity of invasion and minimal bandwidth ordering	283
9.3.4	Example illustrating the invasion algorithm	285
9.3.5	Implementation of the invasion algorithm	285
9.4	The Essex Solution Synthesis Algorithms	287
9.4.1	The AB algorithm	287
9.4.2	Implementation of AB	289
9.4.3	Variations of AB	291
9.4.4	Example of running AB	292
9.4.5	Example of running AP	294
9.5	When to Synthesize Solutions	294
9.5.1	Expected memory requirement of AB	294
9.5.2	Problems suitable for solution synthesis	295
9.5.3	Exploitation of advanced hardware	297
9.6	Concluding Remarks	297
9.7	Bibliographical Remarks	298
<b>Chapter 10</b>	<b>Optimization in CSPs</b>	<b>299</b>
10.1	Introduction	299
10.2	The Constraint Satisfaction Optimization Problem	299
10.2.1	Definitions and motivation	299
10.2.2	Techniques for tackling the CSOP	300
10.2.3	Solving CSOPs with branch and bound	301
10.2.4	Tackling CSOPs using Genetic Algorithms	305
10.3	The Partial Constraint Satisfaction Problem	314
10.3.1	Motivation and definition of the PCSP	314
10.3.2	Important classes of PCSP and relevant techniques	314
10.4	Summary	318
10.5	Bibliographical Remarks	319
	<b>Programs</b>	<b>321</b>
	<b>Bibliography</b>	<b>383</b>
	<b>Index</b>	<b>405</b>

# Figures

Figure 1.1	A possible solution to the 8-queens problem	2
Figure 1.2	Example of a car sequencing problem	4
Figure 1.3	matrix representing a binary-constraint	11
Figure 1.4	Transformation of a 3-constraint problem into a binary constraint	13
Figure 1.5	Example of a map colouring problem	20
Figure 1.6	Example of a scene to be labelled	21
Figure 1.7	The scene in Figure 1.5 with labelled edges	22
Figure 1.8	Legal labels for junctions (from Huffman, 1971)	22
Figure 1.9	Variables in the scene labelling problem in Figure 1.6	23
Figure 1.10	Thirteen possible temporal relations between two events	24
Figure 1.11	Example of a graph matching problem.	27
Figure 2.1	Control of the chronological backtracking (BT) algorithm	36
Figure 2.2	Search space of BT in a CSP	38
Figure 2.3	Search space for a CSP, given a particular ordering	39
Figure 2.4	Searching under an alternative ordering in the problem in Figure 2.3	40
Figure 2.5	Cost of problem reduction vs. cost of backtracking	42
Figure 2.6	A naive solution synthesis approach	45
Figure 3.1	CSP-1: example of a 3-consistent CSP which is not 2-consistent	56
Figure 3.2	CSP-2: example of a 3-consistent but unsatisfiable CSP	64
Figure 3.3	CSP-3: a problem which is satisfiable but not path-consistent	65
Figure 3.4	CSP-4: a CSP which is 1 satisfiable and 3-consistent, but 2-inconsistent and 2-unsatisfiable	67
Figure 3.5	Example of a constraint graph with the width of different orderings shown	72
Figure 3.6	Examples and counter-examples of k-trees	74
Figure 3.7	Relationship among some consistency and satisfiability properties	77
Figure 4.1	Example of a partial constraint graph	85
Figure 4.2	An example showing that running DAC on both directions for an arbitrary ordering does not achieve AC	90
Figure 4.3	Example showing the change of a graph during adaptive- consistency achievement	107
Figure 4.4	A connectionist representation of a binary CSP	111
Figure 4.5	Summary of the input, output and values of the nodes in Guesgen's network	113
Figure 5.1	Example showing the effect of FC	125
Figure 5.2	The control of lookahead algorithms	126
Figure 5.3	Example showing the behaviour of DAC-Lookahead	132

Figure 5.4	Example showing the behaviour of AC-Lookahead	135
Figure 5.5	A board situation in the 8-queens problem	138
Figure 5.6	An example CSP for illustrating the GBJ algorithm	142
Figure 5.7	Variable sets used by Backmark-1	149
Figure 5.8	Example showing the values of <i>Marks</i> and <i>LowUnits</i> during Backmarking	151
Figure 5.9	A board situation in the 8-queens problem for showing the role of Truth Maintenance in a DAC-L and DDBT hybrid algorithm	153
Figure 6.1	Example of a graph and its width under different orderings	159
Figure 6.2	Example illustrating the significance of the search ordering	161
Figure 6.3	The search space explored by BT and FC in finding all solutions for the colouring problem in Figure 6.2(a)	162
Figure 6.4	Example illustrating the steps taken by the Find_Minimal_Width_Order algorithm	165
Figure 6.5	Example showing the bandwidth of two orderings of the nodes in a graph	168
Figure 6.6	Node partitioning in bandwidth determination	172
Figure 6.7	Example showing the space searched by Determine_-Bandwidth	176
Figure 6.8	Example showing the steps taken by Max_cardinality	181
Figure 6.9	Example of a constraint graph in which the minimum width and minimum bandwidth cannot be obtained in the same ordering	183
Figure 7.1	The size of the search space when a problem is decomposable	190
Figure 7.2	Steps for recognizing a 3-tree and ordering the nodes	197
Figure 7.3	Examples of cycle-cutset	202
Figure 7.4	Procedure of applying the cycle-cutset method to an example CSP	205
Figure 7.5	Search space of the cycle-cutset method	206
Figure 7.6	Example illustrating the possible gain in exploiting stable sets	208
Figure 7.7	Search space of the Stable_Set procedure	210
Figure 7.8	Examples of equivalent CSPs and their dual problems	215
Figure 7.9	General strategy underlying the tree-clustering method	216
Figure 7.10	Conceptual steps of the tree-clustering method	223
Figure 7.11	Example showing the procedure of chordal graphs generation	225
Figure 7.12	Example showing the space searched in identifying maximum cliques	228-229
Figure 7.13	Example summarizing the tree-clustering procedure	236
Figure 7.14	Example of a graph and the j-widths of an ordering	239
Figure 7.15	Example of a set of constrained intervals and points and their corresponding temporal constraint graph	242
Figure 7.16	Example of an unsatisfiable temporal constraint graph detected by the AnalyseLongestPaths procedure	246
Figure 7.17	Possible space searched by AnalyseLongestPath for the temporal constraint graph in Figure 7.16	247
Figure 7.18	Some special CSPs and specialized techniques for tackling them	249
Figure 8.1	Possible problems with hill-climbing algorithms	257



Figure 8.2	Example in which the Heuristic Repair Method would easily fail	259
Figure 8.3	Example of a binary CSP and its representation in GENET	263
Figure 8.4	Example of a network state in GENET	264
Figure 8.5	Example of a converged state in GENET	265
Figure 8.6	Example of a network in GENET which may not converge	267
Figure 9.1	The board for the 4-queens problem	277
Figure 9.2	The MP-graph constructed by Freuder's algorithm in solving the 4-queens problem	279
Figure 9.3	Example of an invasion	281
Figure 9.4	Example showing the output of the invasion algorithm	286
Figure 9.5	Constraints being checked in the Compose procedure	290
Figure 9.6	The tangled binary tree (AB-graph) constructed by the AB algorithm in solving the 4-queens problem	293
Figure 10.1	Example of a CSOP	304
Figure 10.2	The space searched by simple backtracking in solving the CSOP in Figure 10.1	305
Figure 10.3	The space searched by Branch & Bound in solving the CSOP in Figure 10.1: branches which represent the assignment of greater values are searched first	306
Figure 10.4	The space searched by Branch & Bound in solving the CSOP in Figure 10.1 when good bounds are discovered early in the search	307
Figure 10.5	Possible objects and operations in a Genetic Algorithm	308
Figure 10.6	Control flow and operations in the Canonical Genetic Algorithm	309



## Notations and abbreviations

Notations	Description	Reference
$\{x \mid P(x)\}$	The set of $x$ such that $P(x)$ is true, where $P(x)$ is a predicate	
$ S $	The size of the set $S$	
$\forall X: P(X):$ $f(X) \equiv Q(X)$	$f(X)$ is defined as $Q(X)$ when $P(X)$ holds; it is undefined otherwise	Chapter 1, footnote 1
$\langle x, v \rangle$	Label — assignment of the value $v$ to the variable $x$	Def 1-2
$\langle x_1, v_1 \rangle \dots \langle x_n, v_n \rangle$	Compound label	Def 1-3
$AC((x, y), CSP)$	Arc $(x, y)$ is arc-consistent in the $CSP$	Def 3-8
$AC(CSP)$	The $CSP$ is arc-consistent	Def 3-9
$CE(S)$	Constraint Expression on the set of variables $S$	Def 2-8
$CE(S, P)$	Constraint Expression on the set of variables $S$ in the $CSP P$	Def 2-9
$C_S$ or $C_{x_1, \dots, x_h}$	Constraint on the set of variables $S$ or $\{x_1, \dots, x_h\}$	Def 1-7
CSP	Abbreviation for Constraint Satisfaction Problem	
$csp(P)$ or $csp((Z, D, C))$	$P$ is a CSP, or $(Z, D, C)$ is a CSP, where $Z$ is a set of variables, $D$ is the set of domains for the variables in $Z$ , $C$ is a set of constraints	Def 1-12
$DAC(P, <)$	The $CSP P$ is directional arc-consistent according to the ordering $<$	Def 3-12

Notations	Description	Reference
$DPC(P, <)$	The CSP $P$ is directional path-consistent according to the ordering $<$	Def 3-13
$D_x$	Domain of the variable $x$	Def 1-1
$G(P)$	The constraint graph of the CSP $P$	Def 1-18
$\text{graph}((V, E))$	$(V, E)$ is a graph, where $V$ is a set of nodes and $E$ is a set of edges	Def 1-15
$H(P)$	The constraint hypergraph of the CSP $P$	Def 1-18
$NC(P)$	The CSP $P$ is node-consistent	Def 3-7
$PC(p, P)$	The path $p$ is path-consistent in the CSP $P$	Def 3-10
$PC(P)$	The CSP $P$ is path-consistent	Def 3-11