



Constraint Weighting Local Search for Constraint Satisfaction

by

John Thornton

B.Bus, Griffith University, Australia (1993)

B.Sc (Hons), Griffith University, Australia (1995)

A thesis submitted in fulfillment
of the requirements of the degree of
Doctor of Philosophy

School of Computing and Information Technology
Faculty of Engineering and Information Technology
Griffith University, Australia

January 2000

Abstract

One of the challenges for the constraint satisfaction community has been to develop an automated approach to solving Constraint Satisfaction Problems (CSPs) rather than creating specific algorithms for specific problems. Much of this work has concentrated on the development and improvement of general purpose backtracking techniques. However, the success of relatively simple local search techniques on larger satisfiability problems [Selman et al. 1992] and CSPs such as the n -queens [Minton et al. 1992] has caused interest in applying local search to constraint satisfaction. In this thesis we look at the usefulness of constraint weighting as a local search technique for constraint satisfaction. The work is based on the clause weighting ideas of Selman and Kautz [1993] and Morris [1993] and applies, evaluates and extends these ideas from the satisfiability domain to the more general domain of CSPs. Specifically, the contributions of the thesis are:

- **The introduction of a local search taxonomy.** We examine the various better known local search techniques and recognise four basic strategies: restart, randomness, memory and weighting.
- **The extension of the CSP modelling framework.** In order to represent and efficiently solve more realistic problems we extend the CSP modelling framework to include array-based domains and array-based domain use constraints.
- **The empirical evaluation of constraint weighting.** We compare the performance of three constraint weighting strategies on a range of CSP and satisfiability problems and with several other local search techniques. We find that no one technique dominates in all problem domains.

- **The characterisation of constraint weighting performance.** Based on our empirical study we identify the weighting behaviours and problem features that favour constraint weighting. We conclude weighting does better on structured problems where the algorithm can recognise a harder sub-group of constraints.
- **The extension of constraint weighting.** We introduce an efficient arc weighting algorithm that additionally weights connections between constraints that are simultaneously violated at a local minimum. This algorithm is empirically shown to outperform standard constraint weighting on a range of CSPs and within a general constraint solving system. Also we look at combining constraint weighting with other local search heuristics and find that these hybrid techniques can do well on problems where the parent algorithms are evenly matched.
- **The application of constraint weighting to over constrained domains.** Our empirical work suggests constraint weighting does well for problems with distinctions between constraint groups. This led us to investigate solving real-world over constrained problems with hard and soft constraint groups and to introduce two dynamic constraint weighting heuristics that maintain a distinction between hard and soft constraint groups while still adding weights to violated constraints in a local minimum. In an empirical study, the dynamic schemes are shown to outperform other fixed weighting and non-weighting systems on a range of real world problems. In addition, the performance of weighting is shown to degrade less severely when soft constraints are added to the system, suggesting constraint weighting is especially applicable to realistic, hard and soft constraint problems.

Contents

1	Introduction	1
1.1	Constraint Weighting for Constraint Satisfaction	1
1.1.1	Constraint Satisfaction	1
1.1.2	Constraint Satisfaction Algorithms	2
1.1.3	Constraint Weighting	3
1.2	Research Problems	4
1.3	Contributions	5
1.4	Outline	6
2	Constraint Satisfaction Techniques	7
2.1	Definitions	7
2.2	Constructive Techniques	9
2.3	Local Search Techniques	11
2.3.1	Restart Strategies	15
2.3.1.1	Local Minimum Random Restart	15
2.3.1.2	Fixed Iteration Restart	15
2.3.1.3	GSAT	16
2.3.1.4	Value Propagation	17
2.3.2	Stochastic Strategies	18
2.3.2.1	Simulated Annealing	18
2.3.2.2	WSAT	20
2.3.3	Memory Strategies	21
2.3.3.1	Tabu Search	21
2.3.3.2	HSAT, NOVELTY and RNOVELTY	23
2.3.4	Weighting Strategies	25
2.3.4.1	Developments in Constraint Weighting	27
2.3.4.2	Constraint Weighting and Tabu Search	28
2.4	Summary	29

3	Modelling Realistic Problems	30
3.1	Specific and General Solutions	30
3.2	Problem Descriptions	31
3.3	Binary vs. Non-Binary Representation	33
3.3.1	Transforming Non-Binary CSPs	33
3.3.2	Domain Size Issues in Non-Binary Transformations	35
3.3.3	Partial Non-Binary to Binary Transformation	36
3.3.4	Defining Constraints for Tupled Domains	38
3.3.5	Lessons for General Problems	40
3.4	Representing Complex Move Operators	41
3.4.1	Making a Move in a Timetabling Problem	41
3.4.2	Defining Array-based Local Search Constraints	43
3.5	Summary	49
4	Constraint Weighting	51
4.1	Background and Motivations	51
4.2	Constraint Weighting Algorithms	53
4.3	WSAT and Tabu Search Algorithms	55
4.4	Experimental Results	56
4.4.1	Satisfiability Results	56
4.4.2	CSP Results	60
4.4.3	Constraint Weight Curves	62
4.4.4	Constraint Trajectories	66
4.4.5	Measuring Constancy	70
4.4.6	Measuring Problem Structure	71
4.5	Analysis	75
4.5.1	Constraint Weighting Behaviour	75
4.5.2	Identifying Hard Constraint Groups	76
4.5.3	Scaling Effects	80
4.5.4	Overall Behaviour	81
4.6	Summary	84

5	Improving Constraint Weighting	85
5.1	Background and Motivations	85
5.2	Hybrid Techniques	86
5.3	Arc Weighting	88
5.3.1	An Efficient Network Representation	89
5.3.2	Modifications to the Weighting Algorithm	91
5.4	Arc Weighting Experimental Results	92
5.4.1	Arc Weighting on Specialised and General Problem Domains	92
5.4.2	Arc Weighting Performance	94
5.5	Analysis of Arc Weighting	95
5.5.1	Distinguishing Moves	95
5.5.2	Arc Weighting Costs	96
5.5.3	Effects of Problem Size	98
5.5.4	Divergence	99
5.5.5	Applicability to Other Domains	99
5.6	Hybrid Experimental Results	100
5.7	Analysis of Hybrid Algorithm Performance	101
5.8	Summary	102
6	Over-Constrained Problems	104
6.1	Background and Motivations	104
6.2	Constraint Weighting for Over-Constrained Problems	106
6.2.1	Weighting with Hard and Soft Constraints	107
6.2.2	Dynamic Constraint Weighting	110
6.3	Experiments	112
6.3.1	Control Algorithms	112
6.3.2	Comparison Algorithms	113
6.3.3	Test Problems	114
6.3.4	Results	116
6.4	Analysis	117
6.4.1	Nurse Rostering	117
6.4.2	Timetabling	119
6.4.3	RLFAPs	120

6.4.4 Overall Comparison	122
6.5 Summary	126
7 Conclusion	127
7.1 Summary	127
7.2 Future Work	130
Appendix: Zero One Block Constraints	133
Bibliography	135

List of Figures

1.1	An example solution to a four queens chess problem	2
1.2	Constraint weighting four queens example	3
2.1	A backtracking algorithm	9
2.2	Thrashing behaviour in backtracking	10
2.3	A general local search algorithm	12
2.4	Hill-climbing version of GenerateLocalMoves	13
2.5	Hill-climbing version of MakeLocalMove	14
2.6	Example local search topologies [Morris, 1993]	15
2.7	Graphical analysis of optimal restart value	16
2.8	GSAT version of GenerateLocalMoves	17
2.9	SA version of GenerateLocalMoves	18
2.10	SA version of MakeLocalMove	18
2.11	WSAT version of MakeLocalMove	19
2.12	WSAT version of GenerateLocalMoves	20
2.13	Tabu search version of GenerateLocalMoves.	21
2.14	RNOVELTY version of GenerateLocalMoves	24
2.15	RNOVELTY version of MakeLocalMove.	25
2.16	Using constraint weighting for graph colouring	25
2.17	Constraint weighting version of GenerateLocalMoves.	26
3.1	Non-binary and binary constraint graphs	34
3.2	A non-binary staff requirement constraint.	35
3.3	Nurse domain values for simplified problem	38
3.4	Example solution for simplified problem	39
3.5	All purpose getCostChange method.	44
3.6	testChange method for alldifferent constraint	44
3.7	Array version of testChange method for alldifferent	45

3.8	A violated block constraint	46
3.9	A satisfied block constraint	46
3.10	Array version of testChange method for block constraint	47
3.11	getBlockLength method for block constraint	47
3.12	An unsatisfied gap constraint	48
3.13	A satisfied gap constraint	48
3.14	getGapLength method for gap constraint	48
3.15	Array version of testChange method for gap constraint	49
4.1	Three strategies for constraint weighting	54
4.2	Result plot for large DIMACS 3-SAT problems.	58
4.3	An example constraint weight curve	62
4.4	Constraint weight curves for various 3-SAT problems.	63
4.5	Constraint weight curves for different constraint weight methods	63
4.6	3-SAT and log function comparison	64
4.7	Non-uniform DIMACS constraint weight curves	64
4.8	Uniform DIMACS constraint weight curves	65
4.9	CSP constraint weight curves	65
4.10	Changing weight order for 4 selected constraints	67
4.11	Weight trajectories of the 24 most heavily weighted AIM 1 constraints	68
4.12	Weight trajectories of the 2 nd 10 most heavily weighted AIM 1 constraints	69
4.13	Weight trajectories of the first 17 most heavily weighted r100 constraints	69
4.14	Constancy measure Ct of the top 10% of the heaviest weighted constraints	70
4.15	Neighbour count ranges as a proportion of random neighbour counts	74
4.16	Neighbour count std deviations as a proportion of random neighbour counts	74
4.17	Roster and timetabling constraint weight curves	78
4.18	Top 5% of roster and timetabling constraint weight curves	79
5.1	NOVELTYWGT version of GenerateLocalMoves	87
5.2	A simple constraint weighting scenario	88
5.3	Arc weight cost function	90
5.4	Arc weight version of GenerateLocalMoves	91
5.5	Proportion of solved problems by time	96

5.6	Proportion of solved problems by iterations	97
5.7	Comparison of hill climbing moves	97
5.8	Number of minima by iterations	98
5.9	Number of solved satisfiability problems by iterations	99
6.1	GenerateLocalMoves for over-constrained constraint weighting	106
6.2	Weighting hard and soft constraints	109
6.3	The Flexible Weight Adjustment algorithm	112
6.4	Nurse rostering anytime curves for weighting algorithms	118
6.5	Nurse rostering anytime curves for comparative algorithms	119
6.6	Timetabling anytime curves for weighting algorithms	120
6.7	Timetabling anytime curves for comparative algorithms	121
6.8	RLFAP anytime curves for weighting algorithms	122
6.9	RLFAP anytime curves for comparative algorithms	122

List of Tables

2.1	A local search taxonomy	28
3.1	An example tupled nurse variable domain.	37
4.1	Results for small 3-SAT problems	57
4.2	Results for structured DIMACS problems.	59
4.3	Results for CSPs	61
4.4	Averaged small world measures for each problem set	72
4.5	Statistics for variable neighbour counts by problem domain	73
4.6	Parameter settings for WSAT and TABU algorithms	83
5.1	Comparison of mean performance values	94
5.2	Table 5.1 ARCWGT values as a proportion of MINWGT values	94
5.3	Comparison of iteration speed	98
5.4	3-SAT results for hybrid weighting algorithms	101
5.5	DIMACS results for hybrid weighting algorithms	101
6.1	Averaged results for 16 nurse rostering problems	117
6.2	Averaged results for 10 random timetabling problems.	120
6.3	Averaged results for 4 RLFAPs (1,2,3 and 11)	121
6.4	Comparison of Chapter 4 and Chapter 6 nurse rostering success rates	123
6.5	Comparison of Chapter 4 and Chapter 6 timetabling success rates	123
6.6	Comparison of original and adapted GLS performance	125

Definitions of Abbreviations and Terms

AIM refers to satisfiability problems created using an AIM generator (named after Asahiro, Iwama and Miyano, see [Asahiro *et al.*, 1993]). The special feature of an AIM generator is that it can build single solution problems.

ARCWGT a local search constraint weighting heuristic that additionally weights constraints that are simultaneously violated at a local minimum.

BEST a stochastic local search heuristic that either moves randomly or selects the best cost move according to a probability or noise level p .

BESTWGT a local search heuristic that combines BEST and MOVEWGT.

bin40 refers to the randomly generated *binary* CSPs used in the thesis with 30 variables, each with 10 domain values, a constraint density of 40% and a constraint tightness of 32%.

bin80 refers to the randomly generated *binary* CSPs used in the thesis with 30 variables, each with 10 domain values, a constraint density of 80% and a constraint tightness of 17%.

CNF Conjunctive Normal Form: CNF problems are made up of a conjunction of clauses of disjunct literals.

CSP Constraint Satisfaction Problem: a CSP is a problem expressed in terms of variables with domain values and constraints that define the allowable combinations of domain values for the variables. A solution to a CSP is an instantiation of all variables such that all the constraints are satisfied.

Ct Constancy measure that looks at the amount of change in the top 10% of weighted constraints during a search.

DIMACS benchmark refers to the set of benchmark satisfiability problems available from the Center for Discrete Mathematics and Computer Science at <ftp://dimacs.rutgers.edu/pub/challenge/sat/benchmarks/cnf>.

DWA Downward Weight Ajustment: a dynamic local search constraint weighting heuristic for hard and soft constraint problems where the hard constraint weight multiplier is initially set to the total number of soft constraints + 1. Hard weight is then adjusted downwards during the search to equal the number of soft constraints violated in the best solution found so far + 1.

- EFLOP** *Escaping From Local Optima by Propagation*: a local search heuristic that uses value propagation to escape from local minima [Yugami *et al.*, 1994].
- FWA** *Flexible Weight Adjustment*: a dynamic local search constraint weighting heuristic for hard and soft constraint problems where the hard constraint weight multiplier is initially set to the weight of a soft constraint + 1. This weight is then incremented at a local minimum if any hard constraints are violated, otherwise it is decremented.
- GLS** *Guided Local Search*: a local search technique developed by [Voudouris and Tsang, 1996] that penalises problem features at a local minimum according to a utility function.
- GSAT** original local search heuristic proposed by Selman *et al.* [1992] for solving satisfiability problems.
- HSAT** a variant of GSAT proposed by [Gent and Walsh, 1993] that breaks ties on equal cost moves by considering when a move was last made.
- k-SAT** refers to satisfiability problems with a fixed number of k literals in each clause, e.g. 3-SAT problems all have 3 literals per clause.
- ii32** refers to the *inductive inference* problems from the DIMACS challenge set used in the thesis (namely ii32b3, ii32c3, ii32d3 and ii32e3).
- MAX** a local search constraint weighting heuristic for hard and soft constraint problems where the hard constraint weight multiplier is fixed to the total number of soft constraints + 1.
- MAX-SAT** refers to over-constrained satisfiability problems where the objective is to satisfy as many clauses as possible.
- MIN** a local search constraint weighting heuristic for hard and soft constraint problems where the hard constraint weight multiplier is fixed to the number of soft constraints violated in an optimal solution + 1.
- MINWGT** a local search constraint weighting heuristic that adds weight to violated constraints at a local minimum.
- MOVEWGT** a local search constraint weighting heuristic that adds weight to a violated constraint when an overall cost improving move that also improves the constraint cannot be found.
- NOVELTY** a stochastic local search heuristic proposed in [McAllester *et al.*, 1997] that evaluates moves based on how recently the move was last made.
- NOVELTYWGT** a local search heuristic that combines NOVELTY and MOVEWGT.

- par** refers to the *parity* function learning problems from the DIMACS challenge set used in the thesis (namely par8-2-c and par8-4-c).
- PCSP** Partial Constraint Satisfaction Problem: a formalism for representing and solving over-constrained problems by searching for a solution that partially satisfies the problem constraints (from [Freuder and Wallace, 1992]).
- r100** (also *r200* and *r400*) refers to randomly generated 3-SAT problems (see *k*-SAT above) with a clause to variable ratio in the cross-over region of 4.3 : 1. *r100* refers to 100 variable problems, *r200* to 200 variable problems, etc.
- RLFAP** refers to Radio Link Frequency Assignment Problems based on the real problem of assigning frequencies to radio links (made available by the French Centre d'Electronique l'Armement at listserver@saturne.cert.fr).
- RNOVELTY** a stochastic local search heuristic proposed in [McAllester *et al.*, 1997] that evaluates moves based on how recently the move was last made and the relative costs of the two most promising moves.
- RNOVELTYWGT** a local search heuristic that combines RNOVELTY and MOVEWGT.
- SA** Simulated Annealing: a stochastic local search heuristic modelled after the physical cooling process of heated atoms [Abramson, 1992].
- ssa** refers to the circuit fault diagnosis problems from the DIMACS challenge set used in the thesis (namely ssa7552-038, ssa7552-158, ssa7552-159 and ssa7552-160).
- TABU** a constraint sampling local search heuristic that avoids undoing recently made moves [Glover, 1989].
- TABUWGT** a local search heuristic that combines TABU and MOVEWGT.
- tt_rand** refers to the randomly generated timetabling problem set used in the thesis where classes are assigned student groups, staff and room requirements on a random basis.
- tt_struct** refers to the randomly generated timetabling problem set used in the thesis that reflects the structure of a realistic problem.
- UTILWGT** a constraint weighting algorithm based on the utility function proposed in [Voudouris and Tsang, 1996].
- WSAT** refers to a family of local search techniques that grew out of the original WalkSat heuristic [Selman *et al.*, 1994] (includes BEST, NOVELTY and RNOVELTY).

Acknowledgments

I would like to thank my supervisor Dr. Abdul Sattar for his tireless support and encouragement, and for always pointing me in the right direction. I would also like to thank Dr. Clyde Wild and the Gold Coast Campus of Griffith University for their generous financial assistance. Finally, I would like to thank Byungki Cha, Paul Morris, Bart Selman, Peter van Beek and Benjamin Wah for sharing their code, helpful comments and correspondence during the process of completing this thesis.

Statement of Originality

This work has not previously been submitted for a degree or diploma to any university. To the best of my knowledge and belief, the thesis contains no material previously published or written by another person except where due reference is made in the thesis itself.

Signed:

January 2000