Technical Report CSM-359,
Department of Computer Science,
University of Essex,
January 2002

CONSTRAINT SATISFACTION IN BUSINESS PROCESSES MODELLING

Edward P.K. TSANG
Department of Computer Science
University of Essex, Wivenhoe Park, Colchester CO4 3SQ, United Kingdom
Tel: +44 1206 872774; fax: +44 1206 872788; email:edward@essex.ac.uk
http://cswww.essex.ac.uk/CSP/
Technical Report CSM-359

SUMMARY

This paper gives an introductory overview of constraint satisfaction and illustrates its relevance to business modelling. Constraint satisfaction involves assigning values, often from finite domains, to a given set of variables, satisfying a set of constraints, which could take arbitrary form. Constraint Satisfaction is a general problem that appears in many commercial applications. The technology has matured and it supports a multi-million Pounds business. In this paper, we provide a brief survey on basic constraint satisfaction techniques. We then use an example to illustrate how business processes can be modelled with constraints. This leads to the definition of the *Open Constraint Satisfaction Problem*, a branch of constraints research. Finally, we set the research agenda by identifying some of the important areas in Open Constrained Optimization research. By m**odelling business problems as constraint satisfaction problems, one may learn from a wide range of techniques from constraint programming research.**

KEY WORDS: constraint satisfaction, constrained optimization, business modelling

1. Introduction – What is Constraint Satisfaction?

Many problems can be cast as *Constraint Satisfaction Problems* [Tsang 1993; Freuder & Mackworth 1994]. A constraint satisfaction problem involves assigning values to variables, satisfying a set of constraints, which may take arbitrary form (such as databases or functions). For convenience, we call the assignment of a value *v* to a variable *x* a *label*, denoted by <*x*, *v*>. A set of labels (one for each distinct variable) together is called a *compound label*, denoted by ($<x_1, v_1> <x_2, v_2> \ldots <x_k, v_k>$). More formally, a *Constraint Satisfaction Problems* is defined as a triple [Tsang 1993]:

$$(Z, D, C)$$

where Z is a set of variables. D is a function that maps each variable in Z to a *domain*, which is a set of values of arbitrary type (such as integers, people, days of the week). In most constraint satisfaction research, domains are finite. C is a set of constraints. Each constraint *c* in C restricts the values that a subset of variables S in Z may take simultaneously; *c* maps each compound label for S to true (satisfied) or false (unsatisfied). An example of a constraint is "*the sum of x, y and z must be an even number*". The task is to find a compound label for all variables in Z, such that all constraints in C are satisfied. Such a compound label is called a *solution tuple*. In some problems, some solutions are better than others. In that case, the constraint satisfaction problem is extended to a constrained optimisation problem:

$$(Z, D, C, f)$$

where (Z, D, C) is a standard constraint satisfaction problem and *f* is a function that maps every compound label to a numerical value. The task is to find a solution tuple that maximizes or minimizes *f*.

Constraint satisfaction is a very general problem. In research, constraint programming plays a central part in logic programming [Marriott 1998]. Many traditional operational research problems, such as the travelling salesman problem, vehicle routing problem, can be cast as constraint satisfaction problems [Voudouris 1997; Kilby et al 2000]. Other example applications are. Constraint programming is a multi-million Pounds business. Systems such as ILOG [Puget 1995] and ECLiPSe [Lever et al 1995] have been applied to a variety of applications, such as resource allocation, car manufacturing, staff timetabling, airline scheduling and satellite tasks scheduling. Constraint technology has been used by companies such as British Telecom, British Airway, French Railway, Cathay Pacific and Port of Singapore.

2. Basic Constraint Satisfaction Techniques, an Overview

A large number of techniques have been invented for constraint satisfaction. In this section, we shall introduce some of them. Constraint-solving techniques can be classified into three categories:

(a) *Problem reduction* – to reduce the given problem to one that is hopefully easier to solve;

(b) *Search* – to actively look for solution tuples, either systematically or stochastically;

(c) *Solution synthesis* – to build up the set of all solutions systematically.

Most work so far falls into the *problem reduction* and *search* categories, which will be elaborated in the following sub-sections.

2.1. Problem Reduction in Constraint Satisfaction

Problem reduction builds on the idea of *constraint propagation*. A value is *redundant* if it is provable that it will appear in no solutions. Problem reduction attempts to achieve the following goals:

(a) To remove redundant values from domains, with the hope to reduce the problem to one that is easier to solve, partly because the remaining problem is smaller in size;

(b) To tighten constraints, so that constraints can be propagated more effectively in the remaining problem

(c) To prove that no solutions exist, hence no more effort should be wasted.

As an example, let us assume that one is given the following constraint satisfaction problem:

Variables: x, y, z

Domains: all three variables have the same domain {1, 2, 3, 4}

Constraints: x < y and y < z.

If we focus on the constraint x < y, we shall see that 4 may be removed from the domain of x. This is because if x takes the value 4, no value in the domain of y would satisfy this constraint. In this case, we say that the label <x, 4> has no *support* from y. Similarly, 1 can be removed from the domain of y as it has no support from x in the constraint x < y. At this point, the domains of the variables are reduced to:

Variables	Domains
x	{1, 2, 3}
y	{2, 3, 4}
z	{1, 2, 3, 4}

The idea of removing unsupported values is simple. However, propagating constraints efficiently is far from straightforward. Following from the above example, the constraint y < z will enable one to remove 4 from the domain of y and both 1 and 2 from the domain of z, leaving one with the following domains:

Variables	Domains
x	{1, 2, 3}
y	{2, 3}
z	{3, 4}

Now that the value 4 has been removed from the domain of y, <x, 3> loses its support from y. Therefore, constraints must be re-checked repeatedly until no more redundant values are found.

The above example shows a problem reduction strategy that maintains a property called *arc-consistency* in the remaining problem. In general, by investing more time in computation, one may detect and remove more redundant values. A large amount of effort has been spent to find algorithms to propagate constraints efficiently, e.g. see [Tsang 1993].

In fact, removing redundant values is not the only way to reduce the problem. Another way of reducing a problem is to *tighten constraints*. For example, from the constraints x < y and y < z together, one may create the constraint x < z – 1. This may be used to tighten any existing constraint between x and z, which may be useful under certain circumstances [Borrett 1998].

Problem reduction alone does not normally produce solutions, though it may help to find solutions more efficiently through searching. In the next section, we shall explain how this could be done.

2.2. Complete Search Methods

The majority of research in constraint satisfaction focuses on *backtracking search*. One variable *x* is picked at a time. Each value *v* is examined in turn, to see if <*x*, *v*> together with all the labels committed to so far satisfy all the relevant constraints. The search backtracks if all values have been examined and concluded nonviable. If needed, the search enumerates all combinations of labels for all variables. Therefore, solutions will be found if they exist, hence these methods are *complete*.

Various techniques have been developed to enhance backtrack search. They exploit the nature of constraint satisfaction [Tsang 1993]. Some of such techniques are described below. It is worth pointing out that (a) this is not meant to be an exhaustive list; and (b) these techniques may work together:

1. Lookahead search

   After committing to each label, the remaining problem is reduced using the principles described above. Problem reduction helps to detect dead-ends early, so as not to waste search effort [Haralick & Elliott 1980]. Lookahead exploits the fact that all constraints are well defined in constraint satisfaction problems. In general terms, more computation leads to more propagation, which leads to earlier detection of dead-end. One has to find a balance between the computation cost and the benefit from problem reduction.

2. Learning-while-searching

   Another class of search strategy is to analyse the culprits when dead-ends are encountered. This allows one to backtrack past irrelevant labels, and revise those labels that lead to the dead-end. In addition, one may record combinations of variable assignments, called "*no goods*", which together are known to lead to dead-ends (e.g. see [Dechter 1990] and *No-good Backmarking* [Richards et al 1995]). Learning in this form helps avoiding re-discovery of no-goods repeatedly. Learning exploits the fact that all choices (which are dictated by the domains) are fixed and known in advance in constraint satisfaction.

3. Heuristics for ordering variables and values

   It is commonly believed that, in constraint satisfaction, search efficiency is significantly affected by the order in which the variables are ordered. One widely used heuristic is to label the variable that has the smallest domain next. This heuristic works especially well with problem reduction, where attempts are made to reduce the domains of the unlabelled variables after committing to each label [Haralick & Elliott 1980]. Search efficiency is also affected by the order in which the values are ordered. Effective value-ordering heuristics often comes from domain knowledge (e.g. preference).

2.3. Stochastic constraint satisfaction methods

A complete search method will find a solution whenever one exists, given enough time. In reality, time is always limited. The *combinatorial explosion problem* prevents complete methods from finding solutions within the time available. Stochastic methods sacrifice completeness for speed [Freuder et al 1995]. Hill-climbing algorithms such as Min-conflict Heuristic Repair [Minton 1992], GSAT [Selman *et al* 1992, 1993] and CLS [Prestwich 2001] have been designed for constraint satisfaction. GENET was a neural network approach specifications designed for constraint satisfaction [Wang & Tsang 1991; Davenport et al 1994].

Hill climbing algorithms may settle in local optima. Meta-heuristic methods, such as Simulated Annealing [Kirkpatrick et al 1983], Tabu Search [Glover 1989], Guided Local Search [Voudouris 1997] and Genetic Algorithms [Holland 1975] were designed for escaping local optima. These are general algorithms which may be applied to a wide range of domains beyond constraint satisfaction. Details of them will not be described here; see, e.g. [Reeves 1993].

3. Constraint-based Business Processes Modelling

3.1. Distributed Constraint Satisfaction, a Business Scenario

Business operations are subject to lots of constraints. For example, in a supermarket business, individual stores have to fill their shelves with goods from various suppliers. Each store keeps its stock level of each item according to its size, share of shelf space, sales forecast, replacement lead-time, etc. The store manager has to negotiate, with some flexibility, with the delivery department on the timing of each delivery. The opening hours and the staffing level are constraints to be taken into consideration. The store manager may also have to negotiate with the suppliers or the central warehouse on the quantity of each item to be delivered. The task of the store manager is to maximize its profit, subject to its constraints. This problem may be formulated as a constraint satisfaction or a constrained optimization problem.

The transportation department, the suppliers and the central warehouse will all have their own constraints and objective functions to optimize. For practical reasons, it is often impractical for each store or department to provide the other parties (whether the management or the suppliers) with full information on its own constraints. This may be due to complexity or the dynamic nature of the situation. Each store, department or supplier has to solve its constraint satisfaction problem without complete knowledge. The problem cannot be optimized centrally.

3.2. Open Constraint Satisfaction

Many practical constraint satisfaction techniques, such as *Forward Checking* [Haralick & Elliott 1980] and *MAC* [Sabin & Freuder 1994], depend on constraint propagation. The problem described above involves variables beyond a constraint problem solver's control. Some constraints are also unknown to the problem solver. We shall define such problems as *Open Constraint Satisfaction Problems* as follows:

(Z, D, C, E)

Here (Z, D, C) is a standard constraint satisfaction problem as defined above. E is a set of external constraints. Each external constraint ε in E applies to a subset of variables S in Z; ε maps a compound label for S to true if this compound label satisfies the external constraint that ε implements. Operationally, checking this constraint may involve communicating with another agent.

3.3. Near-optimal solutions are preferred

Open Constraint Satisfaction Problems can be extended to *Open Constrained Optimisation Problems*:

(Z, D, C, E, *f*)

where *f* maps every compound label to a numerical value. The task is to maximize or minimize *f*.

Many techniques in constraint satisfaction are designed to find solutions that satisfy constraints. These include most complete search methods such as *Forward Checking* [Haralick & Elliott 1980] and MAC [Sabin & Freuder 1994] and stochastic methods such as *GSAT* [Selman et al 1992, 1993] and *GENET* [Wang & Tsang 1991; Davenport et al 1994]. In many business processes modelling problems, including the scenario described above, one has an objective function to optimize. In some domains, if near-optimal solutions are much easier to find, which is normally the case, then one may be prepared to accept near-optimal solutions.

3.4. Open Constraint Satisfaction Models

One could define an *Open Constraint Satisfaction Model* to model the business processes of an agent:

(Z, D, C, E, *f*, Ag, EtA, CP)

where:

- (Z, D, C, E, *f*) is an Open Constrained Optimization Problems
- Ag is a set of *Agents*: some variables in Z are shared with other agents.
- EtA, a function that maps every external constraint ε in E to an agent in Ag; EtA: E $\rightarrow$ Ag.

- *Communication protocol* (CP), which comprises:
    - *Communication timing*: i.e. *when* an agent should communicate with another. For example, a service department (e.g. transportation) may pay a passive role. It would only formulate its Open Constrained Optimization Problem after it receives an order from other departments. Then it will negotiate values for the shared variables (e.g. delivery time).
    - *Constraints resolution protocol*: One must also define the protocol for *resolving constraints*. For example, in some organizations, agent A may have higher priority than agent B in assigning values to their shared variables; agent B may only reject the values proposed by A if they violate hard constraints in B.

The communication protocol may be specified verbally rather than mathematically.

4. Research Areas in Open Constraint Satisfaction

Research in distributed constraint satisfaction started over a decade ago [Prosser 1990; Luo et al 1992]. The importance of constraint satisfaction in agent technology has been recognized for some time [Yokoo et al 1991]. Researchers in the distributed artificial intelligence (DAI) community focus on aspects such as communication; some assume that individual constraint-solvers use standard constraint programming algorithms [Durfee 1999]. In the constraint satisfaction community, Yokoo invented efficient algorithms for distributed constraint satisfaction [Yokoo 1994]. However, he assumed very naïve communication algorithms. Tel [1999] introduced a distributed AC-4 constraint propagation algorithm, with the Support List" removed. The result of this is that agents must broadcast to all other agents, instead of communicating to the relevant agents only.

We argue that a stand-alone constraint solver should be very different from a constraint solver in a distributed environment. The latter does not necessarily know any of the constraints held by other agents. The situation could be more complex if each agent has its own (not necessarily compatible) goals. A constraint solver should, for example, remember the results of previous communications and predict the possibility of certain values being acceptable by other agents. The Open Constrained Optimization

Problem is more complex than a standard Constrained Optimization Problem. In the following sections, we highlight some of the research areas that require special attention.

4.1. Modelling

Modelling is the key to Constraint Satisfaction [Freuder 1999; Borrett & Tsang 2001]. This is no exception for Open Constraint Satisfaction and Open Constrained Optimization. The questions that one must ask are: which agent are we modelling? What should be modelled as variables? What are their domains? What are the constraints? What are the objective functions? What is the communication protocol? Answers to these questions determine whether the problem can be tackled by constraint satisfaction techniques. It also has significant influence on how efficient this problem can be solved, as it is the case for standard constraint satisfaction problems. The definition of the communication protocol is particularly relevant to the management of a company, who would want to use the protocol to help achieving the company goals.

4.2. Taking Communication Cost into Consideration

Open Constraint Satisfaction Problems differ from standard constraint satisfaction problems in the way that checking constraints in E normally involves communication with other systems. Such communications are normally costly in terms of time and therefore should be kept to a minimal.

There may be hard constraints restricting the maximum amount of time/iterations that agents may take to agree on labels. In other occasions, the objective function may be augmented by the time that it takes to arrive at a decision – the more time it takes to arrive at a decision, the higher the cost may be. Research related to such situation can be found in the literature, e.g. Korf's Real-time IDA* [1985].

4.3. Algorithms

In Open Constraint Satisfaction Problems, one cannot fully propagate constraints to all the variables without incurring high communication costs. However, *partial constraint propagation* should still be considered. More research is needed in effective and efficient partial constraint propagation algorithms.

Another practical strategy in the literature is to record combinations of variable assignments, called "*no goods*", which together are known to lead to dead-ends. The application of such strategies, for example

No-good Backmarking [Richards et al 1995], to Open Constraint Satisfaction Problems should be looked at more carefully.

A class of population-based stochastic algorithms called *Estimation Of Distribution Algorithms* (EDAs) [Muelenbein et al 1996; Larranaga & Lozano 2001] has been adopted in many optimization problems. *Ant Colony Optimization* (ACO) methods [Dorigo 1999] could be regarded as an EDA algorithm for graph problems. The application of EDAs to Open Constrained Optimization Problems deserves close investigation.

4.4. Reliability

Through communication, a constraint solver may learn that some values stand better chance of being accepted by other agents. These values should be preferred to those which have been rejected. Freuder and his team have conducted preliminary research in reliability in constraint satisfaction [Eaton et al 1998]. Their work focuses on dynamic problems, where some feasible solutions in the current problem may become infeasible when the problem is changed. Their work may shed some light on this research aspect of Open Constraint Optimization. EDAs mentioned above has been applied to incremental learning [Baluja 1994], which is highly relevant to value-selection in the Open Constrained Optimization Problem here.

5. Conclusion

Constraint satisfaction is a general problem. Constraint programming has found applications in many commercial domains. In this paper, we have defined the Open Constraint Satisfaction Problem and the Open Constrained Optimization Problem, in which some variables and constraints are not within the problem solver's control. By modelling business processes as Open Constrained Optimization Problems, one opens oneself to a huge body of techniques in the constraints literature. We have set the research agenda by identifying some of the important areas in Open Constrained Optimization research.

Acknowledgement

The constraint satisfaction project is partially supported by EPSRC grants GR/H/75275, GR/J/42878, GR/L20122 and GR/M46297 as well as the Research Promotion Fund, University of Essex. Many colleagues and students at University of Essex have contributed to this project. In particular, I would like to thank Chang Wang, Jim Doran, John Ford, Paul Scott, Terry Warwick, Andrew Davenport, Alvin Kwan, Chris Voudouris, James Borrett, Tung-leng Lau, Jin Li, Patrick Mills and Mathias Kern.

References

1. Baluja, S. *Population-Based Incremental Learning: A Method for Integrating Genetic Search Based function Optimisation and Competitive Learning*, Technical Report, Carnegie Mellon University, 1994
2. Borrett, J.E., *Formulation selection for constraint satisfaction problems: a heuristic approach*, PhD Thesis, Department of Computer Science, University of Essex, Colchester, UK, 1998
3. Borrett, J.E. & Tsang, E.P.K., *A context for constraint satisfaction problem formulation selection*, Constraints, Vol.6, No.4, 2001, 299-327
4. Davenport, A., Tsang, E.P.K., Wang, C.J. & Zhu, K., *GENET: a connectionist architecture for solving constraint satisfaction problems by iterative improvement*, Proc., 12th National Conference for Artificial Intelligence (AAAI), 1994, 325-330
5. Dechter, R., *Enhancement schemes for constraint processing: backjumping, learning, and cutset decomposition*, Artificial Intelligence, Vol.41, No.3, 1990, 273-312
6. Dorigo, M., *Ant Colony Optimisation*, in: Corne, D., Dorigo, M., & Glover, F., eds., New Ideas in Optimization. McGraw-Hill, London, 1999.
7. Durfee, E.H., *Distributed problem solving and planning*, in Weiss, G. (ed), Multiagent systems, a modern approach to distributed artificial intelligence, MIT Press, 1999, 121-164
8. Eaton, P.S., Freuder, E.C. & Wallace, R.J., *Constraints and agents, confronting ignorance*, AI Magazine, Vol.19, No.2, Summer 1998, 51-65
9. Freuder, E.C. & Mackworth, A., (ed.), *Constraint-based reasoning*, MIT Press, 1994
10. Freuder, E.C., Dechter, R., Ginsberg, M., Selman, B. & Tsang, E., *Systematic versus stochastic constraint satisfaction*, Panel Paper, in Mellish, C. (ed.), Proc., 14th International Joint Conference on AI, Montreal, Canada, August, 1995, 2027-2032
11. Freuder, E.C., *Modeling: the final frontier*, The First International Conference on The Practical Application of Constraint Technologies and Logic Programming (PACLP), London, April 1999, 15-21

12. Glover, F., *Tabu search Part I*, Operations Research Society of America (ORSA) Journal on Computing 1, 1989, 109-206
13. Haralick, R.M. & Elliott, G.L., *Increasing tree search efficiency for constraint satisfaction problems*, Artificial Intelligence, Vol.14, 1980, 263-313
14. Holland, J.H., *Adaptation in natural and artificial systems*, University of Michigan press, Ann Arbor, MI, 1975
15. Kilby, P., Prosser, P., and Shaw, P., *A comparison of traditional and constraint-based heuristic methods on vehicle routing problems with side constraints*, Constriaints, Kluwer Academic Publishers, Vol.5, No.4, 2000, 389-414
16. Kirkpatrick, S., Gelatt, C.D. Jr. & Vecchi, M.P., *Optimization by Simulated Annealing*, Science, Vol.220, No.4598, May 1983, 671-680
17. Korf, R.E., *Depth-first iterative deepening: an optimal admissible tree search*, Artificial Intelligence, Vol.27, 1985, 97-109
18. Larranaga, P., & Lozano, J. A., *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*, Kluwer Academic Publishers, 2001
19. Lever, J., Wallace, M. & Richards, B., *Constraint logic programming for scheduling and planning*, British Telecom Technology Journal, Vol.13, No.1., Martlesham Heath, Ipswich, UK, 1995, 73-80
20. Li, J. & Tsang, E.P.K, (1999a), Improving technical analysis predictions: an application of genetic programming, *Proceedings of The 12th International Florida AI Research Society Conference*, Orlando, Florida, May 1-5, 1999, 108-112.
21. Li, J. & Tsang, E.P.K, Investment decision making using FGP: a case study, *Proceedings of Congress on Evolutionary Computation (CEC'99)*, Washington DC, USA, July 6-9 1999.
22. Li, J. & Tsang, E.P.K, *Reducing failure in investment recommendations using genetic programming*, Computing in Economics and Finance Conference, Barcelona, July 2000 (revised version submitted to Journal of Computational Economics)
23. Luo, Q., Hendry, P. & Buchanan, I., *A new algorithm for dynamic distributed constraint satisfaction problems*, Research Report KEG-4-92, University of Strathclyde, 1992
24. Marriott, K. & Stuckey, P.J., *Programming with constraints, an introduction*, MIT Press, 1998
25. Minton, S., Johnston, M., Philips, A.B. & Laird, P., *Minimizing conflicts: a heuristic repair method for constraint satisfaction and scheduling problems*, Artificial Intelligence, Vol.58, Nos.1-3, (Special Volume on Constraint Based Reasoning), 1992, 161-205
26. Muelenbein, H., & Paass, G., From Recombination of Genes to the Estimation of Distribution Part 1, Binary Parameter. Lecture Notes in Computer Science 1141, Parallel Problem Solving from Nature, 1996, 178-187
27. Prestwich, S., *Trading completeness for scalability: hybrid search for cliques and rulers*, Proceedings, Third International Workshop on Integration of AI and OR techniques in Constraint

Programming for Combinatorial Optimization Problems (CP-AI-OR-01), Wye, UK, 8-10 April 2001, 159-173

28. Prosser, P., *Distributed asynchronous scheduling*, PhD Thesis, Department of Computer Science, University of Strathclyde, November, 1990
29. Puget, J-F., *Applications of constraint programming*, in Montanari, U. & Rossi, F. (ed.), Proceedings, Principles and Practice of Constraint Programming (CP'95), Lecture Notes in Computer Science, Springer Verlag, Berlin, Heidelberg & New York, 1995, 647-650
30. Reeves, C.R. (ed.), *Modern Heuristic Techniques for Combinatorial Problems*, Blackwell Scientific Publishing, 1993
31. Richards, T., Jiang, Y. & Richards, B., *Ng-backmarking -- an algorithm for constraint satisfaction*, British Telecom Technology Journal, Vol.13, No.1., Martlesham Heath, Ipswich, UK, 1995, 102-109
32. Sabin, D. & Freuder, E.C., *Contradicting conventional wisdom in constraint satisfaction*, Proc., 11th European Conference on Artificial Intelligence, 1994, 125-129
33. Selman, B., Levesque, H. & Mitchell, D., *A new method for solving hard satisfiability problems*, Proc., National Conference on Artificial Intelligence (AAAI), 1992, 440-446
34. Selman, B. & Kautz, H., *Domain-independent extensions to GSAT: solving large structured satisfiability problems*, Proc., 13th International Joint Conference on AI, 1993, 290-295
35. Sweeney, R.J., *Some new filter rule test: Methods and results*, Journal of Financial and Quantitative Analysis, 23, 1988, 285-300
36. Tel, G., *Distrubted control algorithms for AI*, in Weiss, G. (ed), Multiagent systems, a modern approach to distributed artificial intelligence, MIT Press, 1999, 539-580
37. Tsang, E.P.K., *Foundations of constraint satisfaction*, Academic Press, London and San Diego, 1993
38. Tsang, E.P.K. & Voudouris, C., *Fast local search and guided local search and their application to British Telecom's workforce scheduling problem*, Operations Research Letters, Elsevier Science Publishers, Amsterdam, Vol.20, No.3, March 1997, 119-127
39. Voudouris, C., *Guided Local Search for Combinatorial Optimisation Problems*, PhD Thesis, Department of Computer Science, University of Essex, Colchester, UK, May 1997
40. Voudouris, C. & Tsang, E.P.K., *Guided local search and its application to the traveling salesman problem*, European Journal of Operational Research, Vol.113, March 1999, 469-499
41. Wang, C.J. & Tsang, E.P.K., *Solving constraint satisfaction problems using neural-networks*, Proceedings, IEE Second International Conference on Artificial Neural Networks, 1991, 295-299
42. Yokoo, M., Ishida, T. & Kuwabara, K., *Distributed constraint satisfaction for DAI problems*, AAAI Spring Symposium on Constraint-based Reasoning, March, 1991, 191-199
43. Yokoo, M., *Weak-commitment search for solving constraint satisfaction problems*, Proc., 12th National Conference for Artificial Intelligence (AAAI), 1994, 313-318