Extending GENET to Solve Fuzzy Constraint Satisfaction Problems

Jason H. Y. Wong and **Ho-fung Leung**

Department of Computer Science and Engineering
The Chinese University of Hong Kong
Shatin, N.T., Hong Kong
{hywong,lhf}@cse.cuhk.edu.hk

Abstract

Despite much research that has been done on constraint satisfaction problems (CSP's), the framework is sometimes inflexible and the results are not very satisfactory when applied to real-life problems. With the incorporation of the concept of fuzziness, fuzzy constraint satisfaction problems (FCSP's) have been exploited. FCSP's model real-life problems better by allowing individual constraints to be either fully or partially satisfied. GENET, which has been shown to be efficient and effective in solving certain traditional CSP's, is extended to handle FCSP's. Through transforming FCSP's into $0-1$ integer programming problems, we display the equivalence between the underlying working mechanism of fuzzy GENET and the discrete Lagrangian method. Simulator of fuzzy GENET for single-processor machines is implemented. Benchmarking results confirm its feasibility in tackling CSP's and flexibility in dealing with over-constrained problems.

Introduction

A constraint satisfaction problem (CSP) (Mackworth 1977) involves finding an assignment of values to variables satisfying all constraints. Apart from the naive generate-and-test paradigm, two main approaches have been developed for solving CSP's. The first combines pure tree search with various degrees of constraint propagation. To improve the efficiency of search, techniques such as dependency-directed backtracking, value- and variable-ordering heuristics have been employed. Despite their extensive literature, these constructive methods still need to spend too much time on large-scale problems due to their exponential time complexity and thrashing behavior. In the second approach, repair-based methods are used. They consist of generating for all variables an initial, possibly inconsistent assignment. Variables are then repaired until all constraints are satisfied. This approach has efficiently solved problems such as the one million queens problem (Minton *et al.* 1990). Nevertheless, this method can get caught in local minima. By combining the min-conflicts heuristic repair method (Minton *et al.* 1990; 1992) with a learning strategy to escape local minima, Tsang and Wang (1992) proposed GENET, which is a generic neural network approach for solving CSP's with binary constraints.

The traditional CSP framework provides an elegant way to model problems with hard constraints. However, when applied to real-life problems, this framework is sometimes inflexible. Various proposals have been put forward in a direction to extend the original CSP model with soft constraints. Viewing a crisp constraint as the set of tuples for which the constraint holds, fuzzy set theory seems a natural choice as the tool for the extension. Several researchers have exploited this possibility by formalizing the concept of fuzzy constraint satisfaction problems (FCSP's). They suggest branch-and-bound algorithm as a substitution for backtracking tree search. Some existing techniques developed in the context of CSP's (for example, local consistency, value- and variable-ordering heuristics (Dubois, Fargier, & Prade 1993b; 1993a; Ruttkay 1994; Guesgen & Philpott 1995; Meseguer & Larrosa 1997)) are also adapted to boost search performance.

Inheriting the disadvantages from their counterparts in CSP algorithms, branch-and-bound searches are not efficient, either. Some researchers thus put their attention on stochastic algorithms and sacrifice completeness. One attempt is the emergent computation model CCM (Kanada 1995). Though not performing as good as tree search algorithms, CCM's sole dependency on local information lends itself to parallelization. Another attempt by Bowen and Dozier (1996a; 1996b) is the development of FMEHA1 and FMEHA2. Both of them are applications of evolutionary algorithms and have been tested on random FCSP's.

In this paper, we propose fuzzy GENET, a stochastic model for solving FCSP's based on GENET. Benchmarking results of GENET and fuzzy GENET exhibit similar performance on traditional CSP's. With the incorporation of the concept of fuzziness, fuzzy GENET excels GENET in the ability to deal with over-

constrained problems.

This paper is organized as follows. After a short introduction of FCSP's in the next section, the fuzzy GENET model is revealed. We provide sufficient settings for the equivalence between the mechanism of fuzzy GENET and the discrete Lagrangian method (Shang & Wah 1998). Benchmarking results of fuzzy GENET on both CSP's and FCSP's are then presented. Finally, we conclude the paper and give the direction of our future work.

Fuzzy Constraint Satisfaction Problems

A *constraint satisfaction problem* (CSP) is defined as a tuple (Z, D, C^c). Z is a finite set of variables. D is a finite set of domains, one associated with each variable in Z. C^c is a set of constraints. Each k-ary constraint c^c is a crisp relation R^c among the variables of a subset $Z' = \{z_1, z_2, \ldots, z_k\}$ of Z. The *characteristic function* μ_{R^c} of R^c maps from $D_{z_1} \times D_{z_2} \times \ldots \times D_{z_k}$ to $\{0, 1\}$. A returned value of 1 signifies satisfaction and 0 violation. A binary CSP is a CSP with unary and binary constraints only. The assignment of value v to variable z is represented by a *label* v_z. A *compound label* is the simultaneous assignment of values to a set of variables. The goal of a CSP is to find a compound label for all variables in Z that satisfies all the constraints in C^c.

A *fuzzy constraint satisfaction problem* (FCSP) defined as (Z, D, C^f) differs from a CSP in the constraints involved. C^f is a set of fuzzy constraints. Each fuzzy constraint c^f is a fuzzy relation R^f among the variables of the subset $Z' = \{z_1, z_2, \ldots, z_k\}$ of Z. The *membership function* μ_{R^f} of R^f maps from $D_{z_1} \times D_{z_2} \times \ldots \times D_{z_k}$ to $[0, 1]$. μ_{R^f} assigns a *degree of satisfaction* $\alpha_{v_{z_1} v_{z_2} \ldots v_{z_k}} \in [0, 1]$ to each compound label $(v_{z_1} v_{z_2} \ldots v_{z_k})$ for variables in Z'. It is an indication of the extent $(v_{z_1} v_{z_2} \ldots v_{z_k})$ satisfies c^f. If it is equal to 0, $(v_{z_1} v_{z_2} \ldots v_{z_k})$ does not satisfy c^f at all. If it is 1, $(v_{z_1} v_{z_2} \ldots v_{z_k})$ fully satisfies c^f. An intermediate value between 0 and 1 signifies partial satisfaction.

The degree of satisfaction of a fuzzy constraint tells us to what extent it is satisfied. It is equal to the degree of satisfaction of the compound label chosen as the assignment. *Global satisfaction degree* $\alpha_P \in [0, 1]$ of an FCSP P shows its overall satisfaction. It is obtained from an aggregation of the degrees of satisfaction of all the fuzzy constraints by an operator f_α. In his first article about fuzzy set theory, Zadeh (1965) proposed to use min as f_α. Bellman and Zadeh (1970) later coined this as confluence of constraints, acquiring different meanings in different cases. With similar point of view, Zimmermann (1996) pointed out that the choice of appropriate aggregation operator largely depends on the context of the problem one deals with.

Threshold $\alpha_0 \in [0, 1]$ is a user-specified lower bound of the acceptable global satisfaction degree. The goal of an FCSP P is to find a compound label such that $\alpha_P \geq \alpha_0$.

Fuzzy GENET

Network Architecture

Fuzzy GENET is a neural network model for solving *binary* FCSP's. Each *label node* in the network represents one label using the same notation i_y. The state of a label node is either *on* or *off*. If a label node is on, it means that its corresponding assignment is being chosen. A *cluster* is the set of all label nodes that represents the labels of the same variable. A *connection* is placed between every pair of label nodes $(i_y j_z)$ of the two clusters of each binary fuzzy constraint (a *unary* fuzzy constraint is treated as a binary fuzzy constraint on the same variable). A 2-tuple is associated with each connection. The first component of the tuple is the degree of satisfaction of the compound label $(i_y j_z)$ according to the corresponding constraint. The second one is the *weight* $W_{i_y j_z}$ of the connection. Weights are initialized by

$$W_{i_y j_z} = \alpha_{i_y j_z} - 1 \tag{1}$$

The *output* O_{i_y} of a label node i_y depends on its state. It is either 1 for on or 0 for off. The *input* I_{i_y} to i_y is the weighted sum of the outputs of all nodes connected to it. As only one label node in each cluster is permitted to be turned on at any time, every state of the network represents an assignment of a value to each variable in the network.

Convergence Procedure

The initial state of the fuzzy GENET network is *randomly* determined. One label node in each cluster is randomly selected to be turned on. In each convergence cycle, every label node calculates its input *asynchronously in parallel*. The node that receives the maximum input in each cluster will be turned on and the others will be turned off. Since there are only negative connections representing conflicting values, the winner in each cluster represents a value assigned to the corresponding variable with least constraint violations. After a number of cycles, the network will settle in a stable state. In a stable state, if $\alpha_P \geq \alpha_0$, an acceptable solution has been found. Otherwise, the network is trapped in a local minimum.

Care has to be taken when there is more than one winning node in a network update. If none of them is already on since the last update, one node will be *randomly* selected to be turned on. If one of them is already on, it will remain on. This is to avoid chaotic or cyclic wandering of the network states.

Learning Procedure

When fuzzy GENET settles in a local minimum, there are some active label nodes that still receive negative input, indicating that some constraints are violated. This happens because the state update of each cluster is only based on local decision and this does not necessarily lead to a global optimal solution. In such case,

the state update rule would fail to make alternative choices.

To overcome these problems, a *heuristic learning rule* which updates the connection weights is used:

$$W_{i_y j_z}^{new} = W_{i_y j_z}^{old} + O_{i_y}^{old} O_{j_z}^{old} (\alpha_{i_y j_z} - 1) \qquad (2)$$

To see how this rule works, suppose we have a network settled in local minimum. There must exist at least two active label nodes i_y and j_z connected by a negative weight. Also i_y and j_z must have the maximum input in their own cluster. However, their inputs will be reduced after every learning cycle, as long as the state of the network does not change. After sufficient number of cycles, either i_y or j_z will not win the competition in their own cluster. Hence, the state of the network will eventually find its way out of the local minimum.

The overall fuzzy GENET algorithm is shown in figure 1.

randomly turn on a label node per cluster
for each $W_{i_y j_z}$ **do**
  set $W_{i_y j_z} = \alpha_{i_y j_z} - 1$
end for
while $\alpha_P < \alpha_0$ **do**
  for each cluster **do**
    (*asynchronously in parallel*)
    turn on only node with maximum input
    and keep original state if needed
  end for
  if local minimum reached **then**
    for each $W_{i_y j_z}$ **do**
      update $W_{i_y j_z}$:
      $W_{i_y j_z}^{new} \leftarrow W_{i_y j_z}^{old} + O_{i_y}^{old} O_{j_z}^{old} (\alpha_{i_y j_z} - 1)$
    end for
  end if
end while

Figure 1: Fuzzy GENET algorithm $\mathcal{F}$

Fuzzy GENET can be seen as an extension to GENET. One way to model GENET using fuzzy GENET is by adopting the minimum function as the aggregation operator f_α and assigning 1.0 as the threshold α_0. Obviously, the invention of fuzzy GENET broadens the area of application of GENET by modifications conforming to the definition of FCSP.

Mechanism of Fuzzy GENET

Fuzzy GENET can tackle both binary CSP's and FCSP's. However, facing the same question to GENET, its underlying mechanism was not clear. Until recently, Choi and Lee (1998) have exploited the relationship between GENET and the Discrete Lagrangian Method. Their results shed light on our research as it has been mentioned above that GENET is an instance of fuzzy GENET. In this section, we employ a similar technique in search of an explanation of how fuzzy GENET works.

A $0-1$ Integer Programming Formulation of Binary FCSP's

To solve a binary FCSP, a fuzzy GENET network is built as explained in the previous section. The principles behind these rules of construction can be applied in transforming a binary FCSP into a $0-1$ *integer programming problem*. Consider a binary FCSP (Z, D, C^f). For all variables $z \in Z$, we introduce one $0-1$ variable x_{j_z} to represent each label j_z. Each x_{j_z} thus corresponds to the label node j_z in the fuzzy GENET network. They exhibit identical behavior by taking a value of 1 when j is assigned to z and 0 otherwise. The vector $(\dots, x_{j_z}, \dots)$ constitutes all the $0-1$ variables in the transformation and we adopt $\mathbf{x}$ as its representation for convenience. In the fuzzy GENET model, only one label node in each cluster can have output equal to 1. This imposes constraints on the $0-1$ variables:

$$c_z(\mathbf{x}) = 1 \quad \forall z \in Z \qquad (3)$$

where $c_z(\mathbf{x})$ gives the number of x_{j_z}, $j \in D_z$, taking the value of 1.

For each 2-compound label of a binary fuzzy constraint in C^f, a connection is established in the fuzzy GENET network to encode its degree of satisfaction. Analogously, a constraint is introduced for each label $(i_y j_z)$ as follows:

$$g_{i_y j_z}(\mathbf{x}) \equiv a_{i_y j_z}(\mathbf{x}) \sqrt{1 - \alpha_{i_y j_z}} = 0 \qquad (4)$$

$$\text{where} \quad a_{i_y j_z}(\mathbf{x}) = \begin{cases} 1 & \text{if } x_{i_y} = x_{j_z} = 1 \\ 0 & \text{otherwise} \end{cases}$$

The constraint is violated when both $0-1$ variables x_{i_y} and x_{j_z} takes a value of 1 and $\alpha_{i_y j_z} < 1$. Intuitively, this means that the 2-compound label $(i_y j_z)$ is chosen for instantiation but it fails to fully satisfy its corresponding binary fuzzy constraint. The reason for incorporating the factor $\sqrt{1 - \alpha_{i_y j_z}}$ into $g_{i_y j_z}$ will become clear in the following sections.

From the fuzzy GENET algorithm, we can see no explicit optimization of a particular function. So to complete the transformation, we set the objective function to be the constant function $f(\mathbf{x}) \equiv 0$. Finally, the corresponding $0-1$ integer programming problem for a binary FCSP is defined as follows:

$$\min f(\mathbf{x}) = 0$$

$$\text{subject to} \quad g_{i_y j_z}(\mathbf{x}) = 0 \quad \forall (i_y j_z) \in T$$

$$\text{and} \quad c_z(\mathbf{x}) = 1 \quad \forall z \in Z \qquad (\mathcal{IP})$$

where $T = \{(i_y j_z) \mid y, z \in Z, i \in D_y, j \in D_z\}$.

Discrete Lagrangian Method Application

Lagrangian methods are well-known classical methods for solving continuous constrained optimization

problems (Simmons 1975). Recently, Shang and Wah (1998) proposed the Discrete Lagrangian Method (DLM) that works in the discrete space. Applying DLM to problem ($\mathcal{IP}$), the Lagrangian function L is defined as

$$\begin{aligned} L(\mathbf{x},\boldsymbol{\lambda}) &= f(\mathbf{x}) + \sum_{(i_y j_z)\in T} \lambda_{i_y j_z} g_{i_y j_z}(\mathbf{x}) \\ &= \sum_{(i_y j_z)\in T} \lambda_{i_y j_z} g_{i_y j_z}(\mathbf{x}) \end{aligned} \quad (5)$$

where $\boldsymbol{\lambda}$ is a vector of *Lagrange multipliers* $(\ldots, \lambda_{i_y j_z}, \ldots)$. Note that constraint 3 is missing in L. We will explain how it is enforced in the next section.

According to the Discrete Saddle-Point Theorem (Shang & Wah 1998), $\mathbf{x}^*$ is a minimum of ($\mathcal{IP}$) if and only if there exists some Lagrange multipliers $\boldsymbol{\lambda}^*$ such that $(\mathbf{x}^*, \boldsymbol{\lambda}^*)$ constitutes a *saddle point* of the associated Lagrangian function $L(\mathbf{x}, \boldsymbol{\lambda})$. In other words, we can apply a saddle point-seeking algorithm to obtain the minimum of ($\mathcal{IP}$). Based on its definition, $L(\mathbf{x}^*, \boldsymbol{\lambda}) \le L(\mathbf{x}^*, \boldsymbol{\lambda}^*) \le L(\mathbf{x}, \boldsymbol{\lambda}^*)$, a saddle point can be reached by performing a descent in the original variable space of $\mathbf{x}$ and an ascent in the Lagrange-multiplier space of $\boldsymbol{\lambda}$. So we end up with the following difference equations:

$$\begin{aligned} \mathbf{x}^{k+1} &= \mathbf{x}^k - \Delta_{\mathbf{x}} L(\mathbf{x}^k, \boldsymbol{\lambda}^k) \\ \boldsymbol{\lambda}^{k+1} &= \boldsymbol{\lambda}^k + \mathbf{g}(\mathbf{x}^k) \end{aligned} \quad (6)$$

where Δ_x is a *discrete gradient operator* and $\mathbf{g}(\mathbf{x}^k)$ is the vector of constraints $(\ldots, g_{i_y j_z}, \ldots)$.

Following Wah and Shang (1998), we translate equation 6 directly to the *generic discrete Lagrangian algorithm* $\mathcal{G}$ as shown in figure 2. Notice that the algorithm is generic in the sense that several details for a real implementation are left unspecified.

set initial $\mathbf{x}$ and $\boldsymbol{\lambda}$
while $\mathbf{x}$ not a solution, *i.e.*, $L(\mathbf{x}, \boldsymbol{\lambda}) > 0$ **do**
  update $\mathbf{x}$: $\mathbf{x} \leftarrow \mathbf{x} - \Delta_{\mathbf{x}} L(\mathbf{x}, \boldsymbol{\lambda})$
  if condition for updating $\boldsymbol{\lambda}$ satisfied **then**
    update $\boldsymbol{\lambda}$: $\boldsymbol{\lambda} \leftarrow \boldsymbol{\lambda} + \mathbf{g}(\mathbf{x})$
  end if
end while

Figure 2: Generic discrete Lagrangian algorithm $\mathcal{G}$ for solving ($\mathcal{IP}$)

Equivalence Theorem

To construct the fuzzy GENET algorithm $\mathcal{F}$ by the generic discrete Lagrangian algorithm $\mathcal{G}$, we need the following five settings (ψ):

- initial $\mathbf{x}$: $\forall z \in Z, j \in D_z \quad x_{j_z} = O_{j_z}$
- initial $\boldsymbol{\lambda}$: $\forall (i_y j_z) \in T \quad \lambda_{i_y j_z} = \sqrt{1 - \alpha_{i_y j_z}}$
- acceptance condition for solution: $\alpha_P \ge \alpha_0$
- $\Delta_{\mathbf{x}} = \Delta_{\mathbf{x}}^{\mathcal{L}} \equiv (\{\Delta_{\mathbf{x}^z} \mid z \in Z\}, \Omega)$:
  - $\Delta_{\mathbf{x}^z}$**:** *partial discrete gradient operator* that performs steepest descent in the x_{j_z} subspace, $\forall j \in D_z$. It exchanges the values of two variables x_{i_z} and x_{j_z}, $i, j \in D_z$ so that the Lagrangian L is minimized. In other words, its operation does not violate constraint 3. If there are two or more ways for the exchange, the choice follows that of algorithm $\mathcal{F}$.
  - Ω**:** order of application of $\Delta_{\mathbf{x}^z}$'s. The same as the order of variable update in algorithm $\mathcal{F}$.
- condition for updating $\boldsymbol{\lambda}$: local minima in $\mathbf{x}$ space is reached

We call this resultant discrete Lagrangian algorithm $\mathcal{L}$ and it is shown in figure 3. Next, we state without prove two lemmas and the *equivalence theorem* between $\mathcal{L}$ and $\mathcal{F}$.

for $z \in Z$ **do**
  randomly choose $i \in D_z$ and set $x_{i_z} = 1$
  for each $j \in D_z \wedge j \ne i$ **do**
    set $x_{j_z} = 0$
  end for
end for
for each $\lambda_{i_y j_z}$, $(i_y j_z) \in T$ **do**
  set $\lambda_{i_y j_z} = \sqrt{1 - \alpha_{i_y j_z}}$
end for
while $\alpha_P < \alpha_0$ **do**
  for each $z \in Z$ **do**
    (*asynchronously in parallel*)
    update $\mathbf{x}$: $\mathbf{x} \leftarrow \mathbf{x} - \Delta_{\mathbf{x}^z}$
  end for
  if local minimum reached **then**
    update $\boldsymbol{\lambda}$: $\boldsymbol{\lambda} \leftarrow \boldsymbol{\lambda} + \mathbf{g}(\mathbf{x})$
  end if
end while

Figure 3: Discrete Lagrangian algorithm $\mathcal{L}$

Lemma 1 *If currently, (i)* $O_{j_z}^{old} = x_{j_z}^{old} \quad \forall z \in Z, j \in D_z$*, (ii)* $W_{i_y j_z}^{old} = -\lambda_{i_y j_z}^{old} \sqrt{1 - \alpha_{i_y j_z}} \quad \forall (i_y j_z) \in T$*, (iii)* $\mathcal{G}$ *updates* $\mathbf{x}$ *once with* $\Delta_{\mathbf{x}^v}$ *and (iv)* $\mathcal{F}$ *updates the states of label nodes in cluster* v *once, then (i)* $O_{j_z}^{new} = x_{j_z}^{new} \quad \forall z \in Z, j \in D_z$ *and (ii)* $W_{i_y j_z}^{new} = -\lambda_{i_y j_z}^{new} \sqrt{1 - \alpha_{i_y j_z}} \quad \forall (i_y j_z) \in T$.

Corollary 1 *As* v *is only a generic variable, lemma* 1 *is true for all* $z \in Z$*. In other words, if currently, (i)* $O_{j_z}^{old} = x_{j_z}^{old} \quad \forall z \in Z, j \in D_z$*, (ii)* $W_{i_y j_z}^{old} = -\lambda_{i_y j_z}^{old} \sqrt{1 - \alpha_{i_y j_z}} \quad \forall (i_y j_z) \in T$*, (iii)* $\mathcal{G}$ *updates* $\mathbf{x}$ *with* $\Delta_{\mathbf{x}} = \Delta_{\mathbf{x}}^{\mathcal{L}}$ *and (iv)* $\mathcal{F}$ *performs once the convergence procedure, then (i)* $O_{j_z}^{new} = x_{j_z}^{new} \quad \forall z \in Z, j \in D_z$ *and (ii)* $W_{i_y j_z}^{new} = -\lambda_{i_y j_z}^{new} \sqrt{1 - \alpha_{i_y j_z}} \quad \forall (i_y j_z) \in T$.

Corollary 1 states that the equivalence between the state of a fuzzy GENET and the values of the parameters of its corresponding Lagrangian function is preserved if simultaneously $\mathcal{F}$ performs the convergence procedure once and $\mathcal{L}$ updates $\mathbf{x}$ with $\Delta\mathbf{x}$.

Lemma 2 *If currently, (i)* $O_{j_z}^{old} = x_{j_z}^{old} \quad \forall z \in Z, j \in D_z$, *(ii)* $W_{i_y j_z}^{old} = -\lambda_{i_y j_z}^{old}\sqrt{1-\alpha_{i_y j_z}} \quad \forall (i_y j_z) \in T$, *(iii)* $\mathcal{G}$ *updates* $\boldsymbol{\lambda}$ *and (iv)* $\mathcal{F}$ *performs once the learning procedure, then (i)* $O_{j_z}^{new} = x_{j_z}^{new} \quad \forall z \in Z, j \in D_z$ *and (ii)* $W_{i_y j_z}^{new} = -\lambda_{i_y j_z}^{new}\sqrt{1-\alpha_{i_y j_z}} \quad \forall (i_y j_z) \in T$.

Lemma 2 states that the equivalence between the state of a fuzzy GENET and the values of the parameters of its corresponding Lagrangian function is preserved if simultaneously $\mathcal{F}$ performs the learning procedure once and $\mathcal{L}$ updates $\boldsymbol{\lambda}$.

By corollary 1 and lemma 2, it is obvious that if the initial state of a fuzzy GENET is equivalent to the initial values of the parameters of its corresponding Lagrangian function, then $\mathcal{F}$ and $\mathcal{L}$ are equivalent. This fact is stated as the following theorem:

Theorem 1 ***(Equivalence Theorem)*** *If* $\mathcal{G}$ *is instantiated with the five settings* (ψ)*, then it is equivalent to* $\mathcal{F}$*. That is,* $\mathcal{L} \equiv \mathcal{F}$.

Benchmarking Results

As mentioned above, fuzzy GENET is an extension of GENET. To illustrate this point, we have built a fuzzy GENET simulator and test the implementation on both binary CSP's and FCSP's. All the benchmarkings are performed on a SUN SPARCstation 10 model 30 running SunOS 4.1.4. Timing results for fuzzy GENET are the median of 100 runs.

The n-queens Problem

The n-queens problem is to place n queens on an $n \times n$ chess-board so that no two queens attack each other. A GENET simulator is built in this experiment. The timing results of GENET and fuzzy GENET are shown in table 1.

Table 1: Results on n-queens problem

n	GENET	Fuzzy GENET
10	0.06s	0.08s
20	0.48s	0.52s
30	2.09s	2.18s
40	7.27s	9.52s
50	12.49s	13.03s
60	23.04s	23.13s
70	32.03s	30.58s

Obviously, the results in table 1 illustrate that fuzzy GENET is almost as efficient as GENET in solving binary CSP's since fuzzy GENET and GENET are similar in handling crisp binary constraints.

The $n \times (n-1)$-queens Problem

In the $n \times (n-1)$-queens problem (Guan 1994), n queens are placed on an $n \times (n-1)$ chess-board so that there exists at least one pair of queens attacking each other. We define that it is better for any two queens attacking each other to be separated by a greater vertical distance. Formally, the problem can be formulated as follows. There are n variables $\{v_1, \ldots, v_n\}$, each with a domain of $\{1, \ldots, n-1\}$. The degree of satisfaction for the fuzzy constraint $\texttt{noattack}(Q_{i_1}, Q_{i_2})$ that prohibits two queens on row i_1 and row i_2 from attacking each other is 1 if these two queens do not attack each other, and $\frac{|i_1-i_2|-1}{n-1}$ if they do. Therefore, when the vertical distance between two queens increases, $|i_1-i_2|$ and the degree of satisfaction also increase.

The results of fuzzy GENET on the $n \times (n-1)$-queens problem are shown in table 2.

Table 2: Results on $n \times (n-1)$ queens problem

threshold	0.9	0.8	0.7	0.6	0.5
20×19	0.40s	0.13s	0.10s	0.08s	0.05s
30×29	2.63s	0.88s	0.28s	0.22s	0.17s
40×39	5.03s	0.88s	0.33s	0.32s	0.32s
50×49	6.22s	1.70s	0.92s	0.77s	0.65s
60×59	9.28s	3.70s	1.85s	1.33s	1.32s
70×69	16.91s	4.54s	2.49s	2.10s	2.03s

Randomly Generated FCSP's

The suite of random FCSP's provided by Dozier (Bowen & Dozier 1996a; 1996b) can be viewed as a triple (n, d, t) where n is equal to 10 and represents the number of variables in the FCSP as well as the domain sizes. d is the network density and t is the average constraint tightness. Values of d and t are taken from the set $\{0.1, 0.3, 0.5, 0.7, 0.9\}$. Since 5 values can be assigned to d and t , there are 25 classes of randomly generated FCSP's. For each class there are 10 instances, making a total of 250. The truth value of each tuple permitted by a constraint has a randomly assigned value within [0..1].

The results in table 3 show the median truth value obtained (in parentheses) and the median time required in seconds.

Table 3: Results on random FCSP's

t	.1	.3	.5	.7	.9
d=.1	.4(.92)	0.3(.89)	1.1(.85)	3.0(.71)	-(0)
d=.3	.6(.74)	0.8(.58)	1.7(.37)	5.2(.03)	-(0)
d=.5	1.0(.48)	0.9(.35)	1.3(.06)	-(0)	-(0)
d=.7	2.2(.33)	3.7(.16)	4.9(.01)	-(0)	-(0)
d=.9	3.2(.26)	3.9(.12)	-(0)	-(0)	-(0)

Comparing with results presented in (Bowen & Dozier 1996a; 1996b), the truth values obtained are similar. For the instances with tightness t tends to 0.9, it is believed that there are no solutions. Results of Bowen and Dozier (1996b) are not shown because what they counted was the number of evaluations performed by their genetic algorithms.

Conclusion and Future Work

In this paper, we have defined the fuzzy GENET model for solving binary FCSP's. Underlying mechanism of fuzzy GENET has also been shown to be based on the discrete Lagrangian method. Benchmarking results on both CSP's and FCSP's show its advantage over GENET.

The direction of our future work is two-fold. The first is to extend fuzzy GENET to solve *non-binary* fuzzy constraints (parallel to the work in (Lee, Leung, & Won 1995)) and try to give similar proof for its mechanism. The second is to exploit the vast freedom in the choice of parameters of the discrete Lagrangian method and sort out efficient *variants* of fuzzy GENET.

Acknowledgments

We would like to thank Kenneth M. F. Choi and Jimmy H. M. Lee for very useful discussions and inspiration. We would also like to thank Gerry Dozier for providing a set of random FCSP's for benchmarking purpose.

This work is supported by the Croucher Foundation Research Grant CF94/21.

References

Bellman, R. E., and Zadeh, L. A. 1970. Decision-Making in a Fuzzy Environment. *Management Science* 17-B(4):141–164.

Bowen, J., and Dozier, G. 1996a. Solving Randomly Generated Fuzzy Constraint Networks Using Iterative Microevolutionary Hill-Climbing. In *Proceedings of the First International Symposium on Soft Computing in Industry.*

Bowen, J., and Dozier, G. 1996b. Solving Randomly Generated Fuzzy Constraint Networks Using Evolutionary/Systematic Hill-Climbing. In *Proccedings of the Fifth IEEE International Conference on Fuzzy Systems*, volume 1, 226–231.

Choi, K. M. F., and Lee, J. H. M. 1998. A Lagrangian Reconstruction of a Class of Local Search Methods. Technical report, The Chinese University of Hong Kong.

Dubois, D.; Fargier, H.; and Prade, H. 1993a. Propagation and Satisfaction of Flexible Constraints. In Yager, R. R., and Zadeh, L. A., eds., *Fuzzy Sets, Neural Networks and Soft Computing.* Kluwer Academic Press.

Dubois, D.; Fargier, H.; and Prade, H. 1993b. The Calculus of Fuzzy Restrictions as a Basis for Flexible Constraint Satisfaction. In *Proceedings of the Second IEEE International Conference on Fuzzy Systems*, volume 2, 1131–1136.

Guan, Q. 1994. *Extending Constraint Satisfaction Problem Solving with Fuzzy Set Theory.* Ph.D. Dissertation, Technical University of Vienna.

Guesgen, H. W., and Philpott, A. 1995. Heuristics for Solving Fuzzy Constraint Satisfaction Problems. In *Proceedings of the Second New Zealand International Two-Stream Conference Artificial Neural Networks and Expert Systems*, 132–135.

Kanada, Y. 1995. Fuzzy Constraint Satisfaction Using CCM – A Local Information Based Computation Model. In *Proceedings of the Fourth IEEE International Conference on Fuzzy Systems*, volume 4, 2319–2326.

Lee, J. H. M.; Leung, H. F.; and Won, H. W. 1995. Extending GENET for Non-binary CSP's. In *Proceedings of the Seventh International Conference on Tools with Artificial Intelligence*, 338–343.

Mackworth, A. K. 1977. Consistency in Networks of Relations. *Artificial Intelligence* 8(1):99–118.

Meseguer, P., and Larrosa, J. 1997. Solving Fuzzy Constraint Satisfaction Problems. In *Proceedings of the Sixth IEEE International Conference on Fuzzy Systems*, volume 3, 1233–1238.

Minton, S.; Johnston, M. D.; Philips, A. B.; and Laird, P. 1990. Solving Large Scale Constraint Satisfaction and Scheduling Problems Using a Heuristic Repair Method. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, 17–24.

Minton, S.; Johnston, M. D.; Philips, A. B.; and Laird, P. 1992. Minimizing Conflicts: A Heuristic Repair Method for Constraint Satisifaction and Scheduling Problems. *Artificial Intelligence* 58(1–3):161–205.

Ruttkay, Z. 1994. Fuzzy Constraint Satisfaction. In *Proccedings of the Third IEEE International Conference on fuzzy Systems*, volume 2, 1263–1268.

Shang, Y., and Wah, B. W. 1998. A Discrete Lagrangian-Based Global-Search Method for Solving Satisfiability Problems. *Journal of Global Optimization* 12(1):61–100.

Simmons, D. M. 1975. *Nonlinear Programming for Operations Research.* Englewood Cliffs, N.J. : Prentice Hall.

Tsang, E. P. K., and Wang, C. J. 1992. A Generic Neural Network Approach for Constraint Satisfaction Problems. In *Neural Network Applications.* Springer-Verlag. 12–22.

Zadeh, L. A. 1965. Fuzzy Sets. *Information and Control* 8:338–353.

Zimmermann, H.-J. 1996. *Fuzzy Set Theory – And its Applications.* Boston: Kluwer Academic Publishers, third edition.