A Geometric Approach to Anytime Constraint Solving for TCSPs

Yeh Hong-ming, Jane Yung-jen Hsu, and Han-shen Huang

Department of Computer Science and Information Engineering
National Taiwan University, Taipei, Taiwan, R.O.C.

Abstract. Temporal constraint satisfaction problems (TCSPs) are typically modelled as graphs or networks. Efficient algorithms are only available to find solutions for problems with limited topology. In this paper, we propose *constraint geometry* as an alternative approach to modeling TCSPs. Finding solutions to a TCSP is transformed into a search problem in the corresponding n-dimensional space. Violations of constriants can be measured in terms of spatial distances. As a result, *approximate* solutions can be identified when it is impossible or impractical to find exact solutions. A real-numbered evolutionary algorithm with special mutation operators has been designed to solve the general class of TCSPs. It can render approximate solutions at any time and improve the solution quality if given more time. Experiments on hundreds of randomly generated problems with representative parameters showed that the algorithm is more efficient and robust in comparison with the path-consistency algorithm.

1 Introduction

A temporal constraint satisfaction problem (TCSP) is a problem that consists of a set of events and a set of constraints on the time points at which the events occur [1]. To solve a TCSP problem is to find a set of assignments, each of which assigns a time point to an event, such that all constraints are satisfied.

When a solution to a TCSP is not available, either because no solution exists or because the algorithms cannot find it in time, an approximate solution may be a good alternative. In fact, there are many practical TCSP applications in which it is acceptable to violate some of the constraints with reasonable costs. Furthermore, there are cases in which a TCSP contains conflicting constraints.

Constraint satisfaction problems are usually formulated as graphs or networks, with algorithms based on graph theory and search [2]. In this paper, a geometric approach, *constraint geometry,* is proposed to model TCSPs. Using this approach, TCSPs can be represented in an intuitive manner, and the concepts of exact and approximate solutions are straightforward. Besides, an evolutionary algorithm based on constraint geometry is also presented here to support *anytime* problem solving for TCSPs.

2 TCSPs

This section introduces a formal definition of TCSPs, which is more rigorous than the one commonly adopted in the literature [1].

Let $X = \{x_1, x_2, \ldots, x_n\}$ be a finite set of variables on $\mathbf{R}$, the set of real numbers. A constraint c over X defines a function $f : X \to \{true, false\}$ which can be represented in the general form below.

$$k_1x_1 + k_2x_2 + \cdots + k_mx_m \in [a_1, b_1] \cup [a_2, b_2] \cup \ldots \cup [a_n, b_n]$$

where n is a finite positive integer; for $i = 1, 2, \ldots, n$, we have $a_i, b_i, k_i \in \mathbf{R}$ and

$$a_1 \le b_1 < a_2 \le b_2 < \ldots < a_n \le b_n$$

We call the left-hand side of a constraint c the *characteristic function* of the constraint, denoted by $\Delta[c]$; and the right-hand side the *true range* of the constraint, denoted by $\tau(c)$. For example, let c_1 be the constraint $x_1 - x_2 + x_3 \in [2, 4]$, then $\Delta[c_1]$ is $x_1 - x_2 + x_3$ and $\tau(c_1)$ is $[2, 4]$. The set of all a_i's and b_i's is called the *landmark set* of the true range $\tau(c)$.

A TCSP is a finite set of *temporal constraints* over X. Each constraint is either a unary constraint whose characteristic function is of the form x_i, or a binary constraint whose characteristic function is of the form $x_i - x_j$. A solution to a TCSP is a unifier [5] $\sigma = \{x_1 \leftarrow h_1, x_2 \leftarrow h_2, \ldots, x_n \leftarrow h_n\}$, where $h_i \in \mathbf{R}$, such that all constraints are satisfied. If the unifier $\{x_1 \leftarrow h_1, x_2 \leftarrow h_2, \ldots, x_n \leftarrow h_n\}$ is a solution to a TCSP containing only binary constraints, the unifier $\{x_1 \leftarrow h_1 + k, x_2 \leftarrow h_2 + k, \ldots, x_n \leftarrow h_n + k\}$, for any constant k, is also a solution. More detailed definitions can be found in Yeh [8].

3 Constraint Geometry

Constraint-satisfaction problems are often treated as search problems. Therefore, to solve a TCSP is to search for a solution in the problem space [6] defined by the TCSP. The basic idea of *constraint geometry* is to formulate a problem in an Euclidean space. Subsequently, the constraints and solutions to a TCSP become geometric objects in the space.

Given a TCSP P on the set of variables $X = \{x_1, x_2, \ldots, x_n\}$, the variable space of P, denoted by $\nu(P)$, is the Cartesian product of the domains of all variables in X. The variable space of a constraint, denoted by $\nu(c)$, is a Cartesian product of the domains of all variables occuring in c.

3.1 Geometry of Constraints

For any TCSP problem P with n temporal variables, each constraint is bounded by a set of parallel hyperplanes in the n-dimensional Euclidean space $\nu(P)$.

Definition 1. Boundary. *Let c be a constraint in a TCSP P, and $L(c)$ be the landmark set of $\tau(c)$. The boundary of c at a landmark $l \in L(c)$, denoted by $b(c,l)$, is the set*

$$\{p \mid p \in \nu(c) \text{ s.t. } \Delta[c](p) = l\}.$$

The boundaries *of the constraint c, denoted by $B(c)$, is the union of its boundaries at all landmarks, namely*

$$B(c) = \bigcup_{l \in L(c)} b(c,l).$$

For example, consider a constraint $c_1 : x_1 - x_2 \in [2,4]$. Fig. 1 depicts the boundaries $b(c_1,2)$ and $b(c_1,4)$, identified by the lines $x_1 - x_2 = 2$ and $x_1 - x_2 = 4$ respectively. The shaded area consists of points that satisfy c_1.

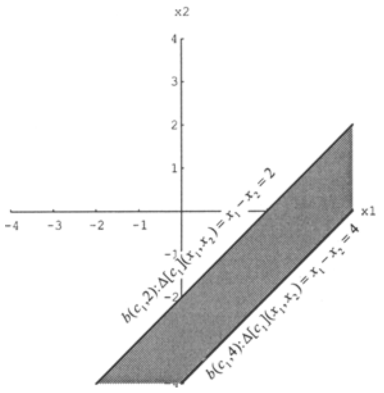

Fig. 1. The boundaries $B(c_1)$ of $x_1 - x_2 \in [2,4]$.

A point p is said to be *inside* the boundaries of c iff $\Delta[c](p) \in \tau(c)$, or *outside* the boundaries otherwise. A point p is *on* the boundaries iff $\Delta[c](p) \in L(c)$. (Note that $L(c) \subset \tau(c)$.) The *distance* between a constraint c and a point p in the space is the shortest distance between p and the boundaries of c if p is outside the boundaries. The distance is zero if p is inside any of the boundaries.

Definition 2. Distance. *Let c be a constraint with $\Delta[c](x_1,x_2) = k_1x_1 + k_2x_2$, and $L(c)$ be the landmark set of $\tau(c)$. Let p be a point in $\nu(X)$. For any landmark $l \in L(c)$, the distance between p and l, denoted by $d(p,l)$, is the minimum of the distance between p and every point in the boundary of c at l. That is,*

$$d(p,l) = \min_{q \in b(c,l)} \{d(p,q)\}$$

where $d(p,q)$ is the Euclidean norm of the vector from p to q. The distance between a point p and the constraint c, denoted by $d(p,c)$, is defined as

$$d(p,c) = \begin{cases} 0 & \textit{if } \Delta[c](p) \in \tau(c), \\ \min_{l \in L(c)}\{d(p,l)\} & \textit{otherwise.} \end{cases}$$

Take the example from Figure 1. Consider the point $(-1,3)$, which is outside the boundaries of c_1, since $\Delta[c_1](-1,3) = -4 \notin \tau(c_1)$. By definition, the distance between $(-1,3)$ and c_1 is based on the boundary at 2, as shown in Figure 2.

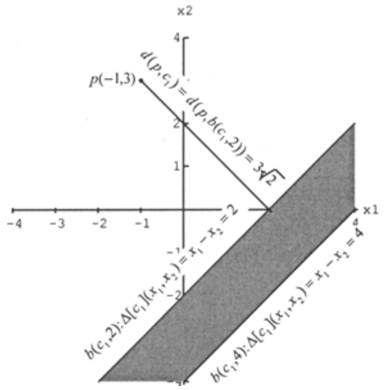
x2
x1
p(−1,3)
d(p,c1) = d(p,b(c1,2)) = 3√2
b(c1,2):Δ[c1](x1,x2) = x1 − x2 = 2
b(c1,4):Δ[c1](x1,x2) = x1 − x2 = 4
-4
-3
-2
-1
1
2
4

Fig. 2. The distance between (-1,3) and constraint c_1.

Given any point $p = (p_1, p_2)$, we have

$$d(p,l) = \frac{|l - p_1 + p_2|}{\sqrt{2}}$$

When c is a unary constraint, the distance equation can be further reduced into

$$d(p,l) = |l - p_1|$$

These equations are very easy to compute, which makes highly efficient implementations possible.

3.2 Geometry of TCSP

Given a problem P, the variable space $\nu(c)$ of any constraint c in P is a subspace of the variable space $\nu(P)$. To manipulate the constraints with respect to the space $\nu(P)$, two operations, *projection* and *inverse projection*, are necessary.

Let S' be a subspace of a variable space S. The *orthogonal projection* (or simply *projection*) of a point p in the space S unto the space S', denoted by

$\pi[S, S'](p)$, is a point in S', say p', that has the same value as p at every corresponding coordinate. The projection of an object o in S unto S' is the union of the projection of every points of o.

The *inverse orthogonal projection* (or simply *inverse projection*) [4] can be defined in terms of projection. The inverse projection of a point p' in the space S' into S, denoted by $\pi^{-1}[S', S](p')$, is the set $\{p \mid p \in S' s.t. \pi[S, S'](p) = p'\}$. In other words, the inverse projection of p' is the union of all points in S whose projection is p'. It should be noted that distances are invariant through inverse projection.

Given a constraint c for a problem P, we can extend any boundary $b(c, l)$ in the subspace $\nu(c)$ to the variable space $\nu(P)$. The boundary of c at l with respect to the variable space $\nu(P)$ is the inverse projection of every point in $b(c, l)$ into $\nu(P)$, i.e.

$$\{\pi^{-1}[\nu(c), \nu(P)](p) \mid p \in b(c, l)\}.$$

For simplicity, it is denoted by $\pi^{-1}[\nu(c), \nu(P)]\{b(c, l)\}$. The boundaries $B(c)$ in $\nu(P)$ is defined in a similar way.

Let us expand the example of c_1 by adding another constraint $c_2 : x_2 - x_3 \in [1, 4]$. Figure 3 shows the boundaries of c_2 in $\nu(c_2)$, which is an $\mathbf{R}^2$ plane.

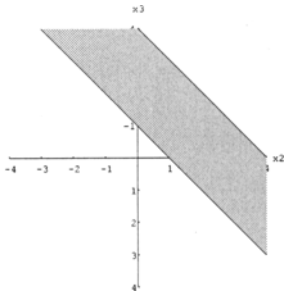

Fig. 3. The boundaries of constraint c_2.

Let $X = \{x_1, x_2, x_3\}$ be the set of variables in the new TCSP P, whose variable space $\nu(P)$ is $\mathbf{R}^3$. To consider c_1 and c_2 in $\nu(P)$, it is necessary to compute the inverse projections of $B(c_1)$ and of $B(c_2)$ into $\nu(P)$. Figure 4 depict $\pi^{-1}[\nu(c_1), \nu(P)]\{B(c_1)\}$ and $\pi^{-1}[\nu(c_2), \nu(P)]\{B(c_2)\}$.

Therefore, any solution to a given TCSP corresponds to a point that is inside the boundaries of all the constraints in its variable space. For instance, for the sample TCSP P presented earlier, a point p is a solution if both $d(p, c_1) = 0$ and $d(p, c_2) = 0$, as shown in Figure 5.

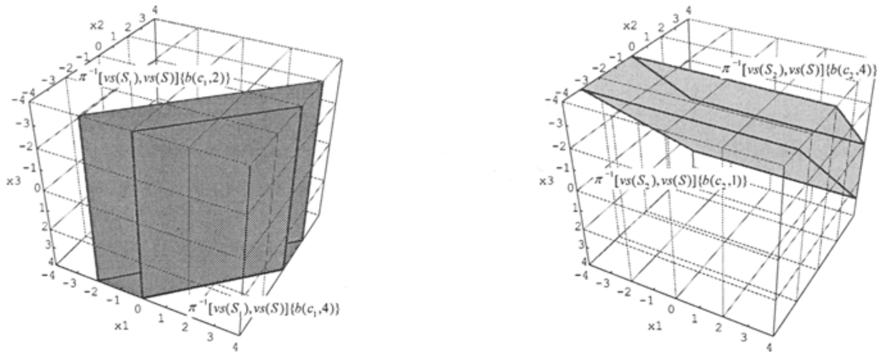
$\pi^{-1}[vs(S_1),vs(S)]\{b(c_1,2)\}$
$\pi^{-1}[vs(S_1),vs(S)]\{b(c_1,4)\}$
$\pi^{-1}[vs(S_2),vs(S)]\{b(c_2,4)\}$
$\pi^{-1}[vs(S_2),vs(S)]\{b(c_2,1)\}$
x1
x2
x3

Fig. 4. Inverse projections into $\nu(P)$.

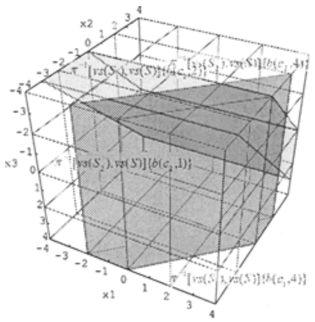

Fig. 5. Solution region for the sample TCSP.

4 Anytime Constraint Solving

Given a TCSP, looking for a solution in its variable space is a daunting task. A good search method should attempt to approach the regions inside the boundaries of all constraints. Intuitively, searching for a solution involves a process of globally minimizing the total potential energy as defined by the distances from the constraints. This section presents an anytime search algorithm, called *Evolutionary Constraint Optimizer* (or ECO), which is a real number genetic algorithm that utilizes the spatial properties of constraint geometry to solve general TCSPs.

4.1 Preprocessing

There are two preparatory tasks before ECO applies genetic algorithms to solve a bounded TCSP: transforming the problem into a canonical form and constricting the domains of the variables. Canonicalization is necessary for the mutation operators described later in this section. The canonicalization procedure is the following:

1. Define a partial order ¡ on the variables such that for any two distinct variables x_i and x_j, if a constraint involving the two variables is in P, then either $x_i < x_j$ or $x_j < x_i$, but not both. If more than one partial order is possible, choose one arbitrarily.
2. Canonicalize binary constraints: For every binary constraint $x_i - x_j \in Q$ in P, if $x_i < x_j$, replace the constraint with an equivalent constraint $x_j - x_i \in -Q$, where $-Q$ is $\{-z \mid z \in Q\}$. For example, if P contains a constraint $x_1 - x_2 \in [1, 2]$ and ECO determines that $x_1 < x_2$, then it is replaced with a new constraint $x_2 - x_1 \in [-2, -1]$.
3. Given constraints with the same characteristic function, ECO will merge them by taking the intersection of their true ranges.

Constricting variable domains helps identify a more reasonable range for the genetic algorithms to explore. For any variable x_i, ECO determines its minimum value $l(x_i)$ and maximum value $u(x_i)$ among all solutions of P. The details of the method can be found in Yeh [8].

4.2 Genetic Encoding

Every individual in ECO consists of a single chromosome representing a possible solution of P. Each gene on the chromosome corresponds to a variable of P: the x_1-gene, the x_2-gene, and so on. Therefore, the number of genes in a chromosome is the same as the number of variables. For ease of presentation, we use the same symbol for a variable and its corresponding gene.

The allele value for a gene g may be either *independent* or *dependent.* The structure of an independent allele is a specific number representing the value assigned to the corresponding variable in P. The strucutre of a dependent allele is a 3-tuple, (h, v, δ), where h is either 0 or another gene, v is a closed interval, and δ is a number indicating an *offset.* Each 3-tuple should satisfy the following conditions:

- The canonicalized problem P contains a constraint c with the characteristic function $\Delta[c] = g - h$.
- The interval v is a component of its true range $\tau(c)$.

The TCSP variable corresponding to g is assigned the value of h plus δ, or simply δ if h is 0. For example, given a TCSP with $x_1, x_2, x_3 \in \mathbf{R}$ and the following constraints (with the partial order $x_1 < x_2 < x_3$):

$$\begin{cases} x_1 \in [-1, 0.18] \cup [0.63, 0.97] \\ x_2 \in [-1, -0.96] \cup [-0.72, -0.49] \cup [0.4, 1] \\ x_3 \in [-0.71, -0.53] \cup [0.31, 0.79] \\ x_2 - x_1 \in [-0.59, -0.36] \cup [0.01, 1] \\ x_3 - x_1 \in [-1, -0.27] \cup [0.35, 0.58] \cup [0.7, 1] \\ x_3 - x_2 \in [-1, -0.72] \cup [-0.25, 0.69] \end{cases}$$

Figure 6 illustrates a chromosome consisting of two dependent alleles and one independent allele.

x_1	$(0,[0.63,0.97],0.72)$
x_2	0.33
x_3	$(x_1,[0.35,0.58],0.4)$

Fig. 6. A chromosome with two dependent alleles.

The allele value of the x_1-gene depends on 0 by an offset of 0.72, thus the value of x_1 is $0 + 0.72 = 0.72$. The allele value of the x_2-gene is independent, thus the variable x_2 is assigned the value 0.33. The allele value of the x_3-gene is dependent on x_1 by an offset of 0.4, so the allele value of the x_3-gene is $0.72 + 0.4 = 1.12$. In other words, this chromosome represents the unifier $\{x_1 \leftarrow 0.72, x_2 \leftarrow 0.33, x_3 \leftarrow 1.12\}$.

4.3 Genetic Operators

ECO adopts the uniform crossover operator [7]. In addition, three mutation operators are defined in ECO: the α-mutation, the β-mutation, and the γ-mutation. The first two apply to genes having either independent or dependent alleles, while the γ-mutation is specially designed for genes with dependent allels only.

The α-Mutation The α-mutation allows ECO to explore the neighborhood of the current individuals. An α-mutated gene has an independent allele. The new allele value is decided by a random variable whose probability distribution is a combination of two normal distributions. More precisely, let x be the gene being α-mutated, $l = l(x)$, $u = u(x)$, and μ be the interpretation of the allele value of x before mutation. The probability density function $\alpha(z; \mu, l, u)$ [3] of the random variable is shown in Equation (1).

$$\alpha(z; \mu, l, u) = \begin{cases} \dfrac{exp(\frac{-(z-\mu)^2}{2(l-\mu)^2})}{\sqrt{2\pi}|l-\mu|} & \text{if } z < \mu \neq l \\ \dfrac{exp(\frac{-(z-\mu)^2}{2(u-\mu)^2})}{\sqrt{2\pi}|u-\mu|} & \text{if } z > \mu \neq u \end{cases} \tag{1}$$

Figure 7 shows a sample case of α with $\mu < l$, that is, the interpretation of the original allele value falls outside of the constricted range. Note that by the definition of α-mutation, the farther μ strays from the range between l and u, the more likely for the new allele value to be outside that range. The rationale is that if an individual with a stray allele is reproducing, bearing the straying allele is possibly advantageous for the individual. This is particularly important when ECO faces problems with conflicting constraints.

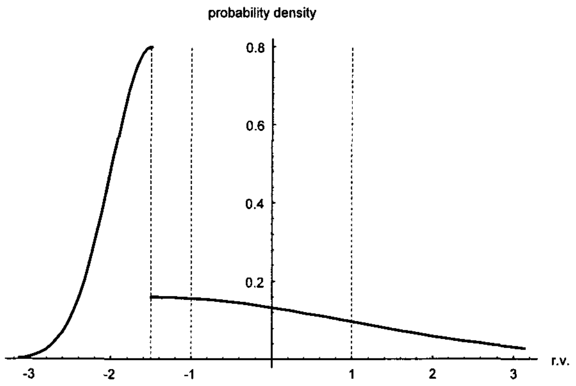

Fig. 7. The probability density function α with $\mu = -1.5, l = -1$, and $u = 1$

The β-Mutation The β-mutation enforces an individual to satisfy a certain constraint. Contrary to the α-mutation, the β-mutation turns the subject gene into having a dependent allele (h, v, δ). Let x be the gene being β-mutated, and $\mathcal{B}$ be the set of all constraints that have x as the first term of their characteristic functions in P. For each constraint c in $\mathcal{B}$, let $y(c)$ be the TCSP variable in the second term of $\Delta[c]$, or 0 if c is a unary constraint; let $K(c)$ be the set of components of $\tau(c)$. We can construct a set Z as

$$Z = \bigcup_{c \in B} [\{y(c)\} \times K(c)]$$

Choose a random member e from Z with a uniform probability distribution. Now h and v in the new allele (h, v, δ) are decided: h is assigned the first element of e, which represents the variable symbol, or 0; and v is assigned the second element of e, the closed interval. The value of δ is then decided by a random variable with a uniform probability distribution over the interval v.

The γ-Mutation The γ-mutation is very similar to the α-mutation, except that it applies to genes with dependent alleles only and it results in a dependent allele. Assume that a gene having allele value $(h, [p, q], \delta)$ is γ-mutated. The only change is its δ value, which is decided by a random variable with the probability density function $\alpha(z; \delta, p, q)$.

4.4 Measurement of Fitness

The fitness measurement f is defined in terms of the distances between the potential solution corresponding to an individual and every constraint of the given TCSP.

Let i be an individual representing the unifier

$$\sigma = (x_1 \leftarrow a_1, x_2 \leftarrow a_2, \ldots, x_n \leftarrow a_n)$$

and $p = (a_1, a_2, \ldots, a_n)$ be a point in the space. The fitness value of i is defined as

$$f(i) = exp(-\sqrt{\Sigma_{c \in C} d(p,c)^2})$$

where $d(p, c)$ denotes the distance between the point p and the constraint c of the TCSP C. Since $0 \leq d(p, c) < \infty$ for each c, it follows that $0 < f(i) \leq 1$, and $f(i) = 1$ iff each of the distances is equal to zero, that is, σ is a solution of P.

4.5 Experiment Results

In order to test the performance of ECO, we performed three sets of experiments, examining the influence of the following factors on the performance of ECO: the *number of variables*, *mean dispersion*, and *constraint density*. The experiments used samples generated from a generic random CSP generator [8] based on a well distributed set of parameters.

The mean dispersion of a TCSP is the algebraic average of the dispersion of each constraint in the problem. The dispersion of a constraint c is the number of intervals that $\tau(c)$ contains. The constraint density of an n-variable TCSP is the number of constraints it has divided by the maximal number of constraints an n-variable TCSP may have.

Number of Variables The goal of the first set of experiments is to test how the number of variables affects the performance of ECO. The experiments tested 70 sample problems, each having 5 to 15 variables. The problems were divided into 7 groups; each group consisted of 10 problems that have the same number of variables. The mean dispersion was 1.5. Figure 8 contrasts the time required by ECO to find an exact solution to the theoretically estimated $O(n^3)$ time required by conventional algorithms based on Floyd-Warshall's algorithm.

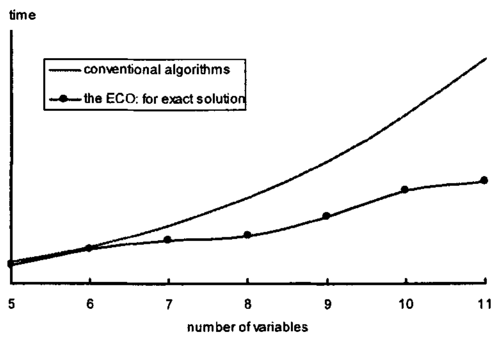

Fig. 8. Time required vs. number of variables

Mean Dispersion The second set of experiments shows how mean dispersion affects the performance of ECO. The experiments tested 150 sample problems with the mean dispersion between 1 and 15. The problems were divided into 15 groups; each group consisted of 10 problems with the same mean dispersion. Each sample problem had 5 variables and 10 constraints. Figure 9 contrasts the time required by ECO to find an individual exact solution to that by conventional algorithms, assuming that path consistency algorithms reduced the mean dispersion by 0%, 50%, 80%, and 95%, respectively.

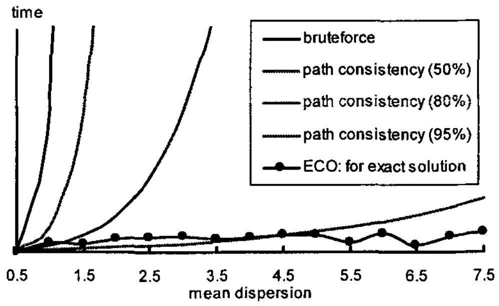

Fig. 9. Time required vs. mean dispersion

Constraint Density The third set of experiments tested 80 sample problems, each having 9 variables, with constraint density from 0.3 to 1. The problems were divided into 8 groups; each group consisted of 10 problems with the same constraint density. Figure 10 contrasts the average time required by ECO to find an individual whose fitness value ≥ 0.8, ≥ 0.9, and $= 1$ (that is, an exact solution) of the sample problems in each of the 8 groups to the (theoretically estimated) time required by conventional algorithm with 50% path consistency and that by the brute-force algorithm.

5 Conclusion

This paper proposes using *constraint geometry* to model TCSPs. The problem of finding a solution becomes a search problem in n-dimensional Euclidean space. An evolutionary algorithm, ECO, has been developed to find exact or approximate solutions. Implementation based on the constraint geometry is relatively straightforward and efficient. Our experiments showed that the time required to find exact solutions is usually much less than exponential (worst case time). The time needed to find good approximate solutions is generally much shorter than finding the exact solutions. Moreover, ECO can return an answer whenever requested, and the quality of the solution improves over time.

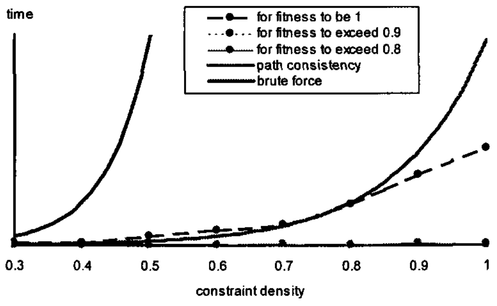

Fig. 10. Time required vs. constraint density

References

1. R. Dechter, I. Meiri, and J. Pearl. Temporal constraint networks. *Artificial Intelligence*, 49:61–95, 1991.
2. R. Dechter and J. Pearl. Network-based heuristic for constraint-satisfaction problems. *Artificial Intelligence*, 34:1–38, 1987.
3. J. L. Devore. *Probability and Statistics for Engineering and the Sciences*. Brooks / Cole Publishing Company, Monterey, California, second edition, 1987.
4. R. M. Haralick and L. G. Shapiro. *Computer and Robot Vision*, page 61. Addison-Wesley, 1993.
5. J. W. Lloyd. *Foundations of Logic Programming*. Springer-Verlag, second, extended edition, 1984.
6. E. Rich and K. Knight. *Artificial Intelligence*, chapter 2. McGraw-Hill, second edition, 1991.
7. G. Syswerda. Uniform crossover in genetic algorithms. In J. D. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, San Mateo, California, 1989. Kaufmann Publishers.
8. H. Yeh. Genetic algorithms for generalized temporal-constraint-satisfaction-problem. Master's thesis, National Taiwan University, June 1995.