ISE-TR-85-49

CONSISTENT LABELING METHODS USING CONSTRAINT NETWORKS

by

Seiichi Nishihara

Katsuo Ikeda

February 25, 1985

Consistent Labeling Methods Using Constraint Networks

Seiichi Nishihara

Katsuo Ikeda

Institute of Information Sciences and Electronics
University of Tsukuba
Sakura-mura, Niihari-gun, Ibaraki 305 Japan

Abstract

Problems of finding totally consistent interpretations when an object consisting of many parts and their locally legal interpretations are given are found in several areas such as image understanding, artificial intelligence. These are a kind of search problems called consistent labeling problems, most of which are NP complete. There are two principal strategies for the consistent labeling problem(CLP): the depth-first approach typified by backtracking and the breadth-first approach typified by constraint propagation. This paper proposes a novel algorithm that takes the breadth-first approach. First, it is shown that any CLP can equivalently be expressed by a constraint network. Then, a general procedure, which repeats relaxing the network and joining two nodes in the network, is proposed. Finally, we introduce three types of concrete algorithms with different levels of heuristics and compare their efficiency and characteristics experimentally.

1. Introduction

One of the approaches to the problem of analyzing or understanding an object composed of many parts is to find a set of labels, each of which gives an interpretation of the corresponding part, satisfying every locally legal combination of interpretation. This kind of problem, which is found in several different areas such as scene labeling in image processing[1], understanding line drawings in artificial intelligence[2], finding homomorphic subgraphs[3] and puzzles like N-queens problem[4], is called in many ways, e.g. the consistent labeling problem (called hereafter CLP for short)[5,6], the relational consistency problem[7], the constraint satisfaction problem[8], the satisficing assignment problem[9] et al.

The algorithms of solving CLP, which is generally an NP complete problem, are mainly classified into two methods: the depth-first approach based upon backtracking and the breadth-first approach typified by constraint propagation. In the former approach, many techniques to resolve thrashing phenomenon caused by wasteful repetition of backtrackings have been proposed, examples of which include forward-checking[4,5], back-marking[4,9] and looking-ahead[4,5]. In the latter approach also, we have some techniques like discrete relaxation[1] and constraint propagation working on a network representing binary relations[1,7,8,10]. Most of those techniques of the breadth-first approach, however, are used not to get the final result but to reduce the number of branches in depth-first algorithm.

One of the earliest breadth-first methods which attains final solutions is the one proposed by Freuder[11], which introduces a constraint network having ability to represent more-than-2-ary relations. In Freuder's method, the network is intially a set of isolated nodes corresponding uniquely to parts, or units. To each node a set of possible interpretations, or labels, is assigned. First, every pair of units, which becomes a new node, is added to the network, to each of which all the possible pairs of labels consistent with its local constraint is also attached. Then, increasing the cardinality one by one, every combination of units is added to the network to create a new node with which a set of legal combinations of labels, or labelings, associated. Finally the node corresponding to the total set of units is added, and all of the solutions are found in the set of labelings attached to this final node. Freuder's method is able to deal with relations with more than two dimensionality; however, one node is inevitably added to the network for every combination of units, and this may cause space and time inefficiency.

This paper proposes a novel filtering algorithm that makes use of the constraint propagation technique. First, it is shown that every CLP can equivalently be expressed by a constraint network in Section 2. Then, in Section 3, an efficient algorithm, which repeats relaxing the network and joining two adjacent nodes in the network, is proposed. Finally, the efficiency of several heuristics are evaluated experimentally in Section 4.

2. The Consistent Labeling Problem and its Equivalent Expression

2.1 The Problem

Here we give a definition of CLP, which is a generalized version of the one given by Haralick and Shapiro[5]. A CLP is represented by a quadruple (U,L,T,R), where U is a set of <u>units</u>, {1,...,M}, and L is a set of <u>labels</u>. Labels are usually possible interpretations, meanings or values being asssigned to the units. T is a set of tuples of units, i.e. $T \subseteq \bigcup_{1 \le i \le M} U^i$. Each tuple t in T tells the units composing t mutually constrain one another. And all of the permitted or legal labelings for t is explicitly given by a |t|-ary relation of labels, R_t ($\subseteq L^{|t|}$), which is called a <u>label constraint relation</u>. |t| is the dimensionality of tuple t. And, $R=\{R_t \subseteq L^{|t|} \mid t \in T\}$.

Solving a CLP is to find all consistent labelings $\lambda=(l_1,\ldots,l_M)$ of units (1,...,M) satisfying $\forall t(\in T)(\lambda(t) \in R_t)$, where $\lambda(t)$ is the projection $(l_{u_1},\ldots,l_{u_{|t|}})$ of λ on $t=(u_1,\ldots,u_{|t|})$. We give an example of CLP.

[Example]

$U=\{1,\ldots,5\}$, $L=\{a,\ldots,e\}$,

$T=\{t_1(=(1,2)),\ t_2(=(2,3,4)),\ t_3(=(4,5)),\ t_4(=(1,3,5))\}$,

$R=\{R_1,R_2,R_3,R_4\}$,

$R_1=\{(a,c),(b,a),(b,e),(c,d),(d,a),(e,b)\}$,

$R_2=\{(a,a,c),(a,c,d),(b,b,a),(b,d,c),(c,a,b),(d,e,a)\}$,

$R_3=\{(a,b),(b,c),(b,d),(c,a),(d,b),(e,a)\}$,

$R_4=\{(a,d,b),(a,e,b),(b,c,b),(c,a,b),(d,c,c)\}$,
where R_i stands for R_{t_i}.

2.2 The Constraint Network

Here we introduce the constraint network providing an equivalent expression of a CLP. The constraint network is defined by an undirected graph (V,E) with a function w which gives an integer value, or a weight, to each arc. With each node i in V, a tuple of units, t_i, and its label constraint relation, R_{t_i} $(\subseteq L^{|t_i|})$ are associated. For any two node i and j there exists an arc (i,j) iff there is at least one common unit possessed by both tuples t_i and t_j. A constraint network equivalent to the given CLP is derived by letting the node set V be the unit relation T. Fig. 1 shows the equivalent constraint network representing the CLP given in the previous section.

Our definition of constraint networks differs from the conventional ones[10,11] in which each node corresponds to a single unit and only binary relations are permitted. However, in the constraint networks proposed above, each tuple in T corresponds to a node; thus, relations with arbitrary dimensions can straightforwardly be expressed. The local constraint required according to each edge is that the labelings associated to the common units appearing in both nodes (i.e. tuples) connected each other by an arc are to be equivalent.

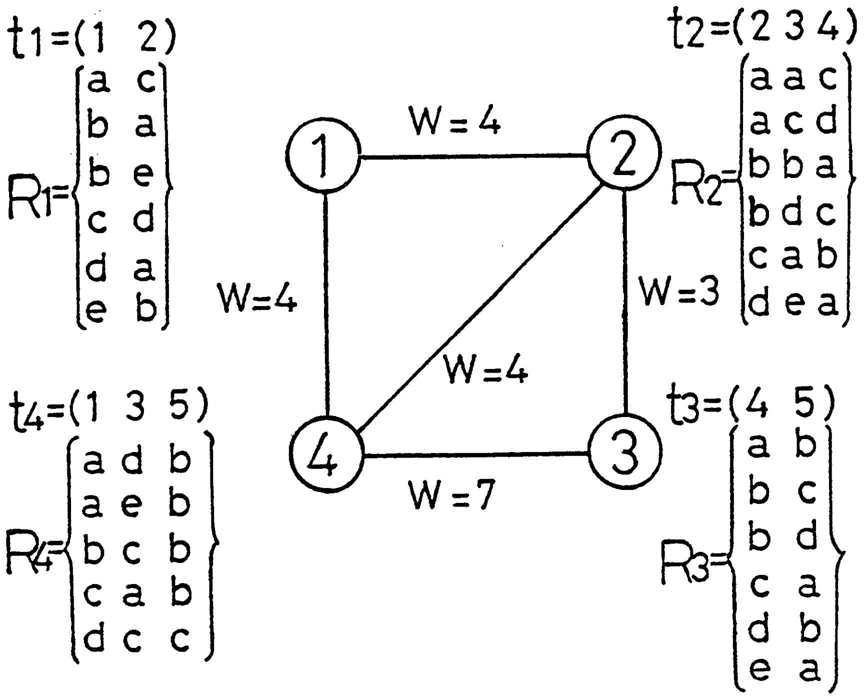

t1=(1 2)
R1={a c, b a, b e, c d, d a, e b}
t2=(2 3 4)
R2={a a c, a c d, b b a, b d c, c a b, d e a}
t3=(4 5)
R3={a b, b c, b d, c a, d b, e a}
t4=(1 3 5)
R4={a d b, a e b, b c b, c a b, d c c}
1
2
3
4
W=4
W=4
W=3
W=4
W=7

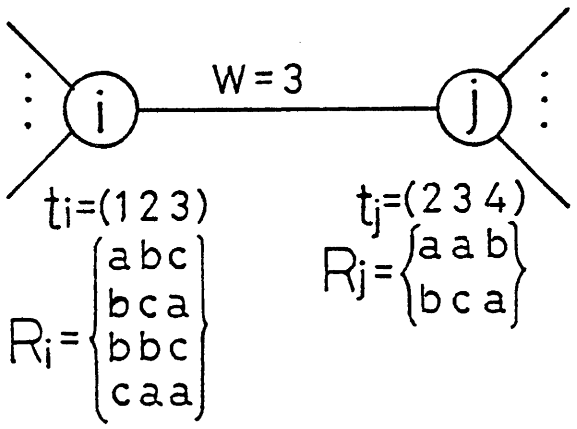
Fig.1 A constraint network.

2.3 The join-operation

Let R_i and R_j be the label constraint relations giving the possible labelings for tuples t_i and t_j, each of which is associated with nodes i and j, respectively. Assume there is an arc (i,j). Then the set of common units $d=\overline{t}_i \cap \overline{t}_j$ is non-empty. Let $r_i[d]$ and $r_j[d]$ be the projection of r_i $(\in R_i)$ and r_j $(\in R_j)$ according to the common units d. Then the join operation of nodes i and j, expressed by join(i,j), deletes the nodes i and j, and creates a new node, say k, to which a tuple t_k and its label constraint relation R_k is assigned, where t_k is the (extended) natural join[12] of R_i and R_j with respect to d. The newly created node k is called the joined node, and R_k is called the joined label constraint relation. The weighting function w assigns each arc (i,j) a weight that is the size of the joined label relation R_k which is to be obtained after joining nodes i and j.

Fig. 2(a) shows an example of a part of a constraint network. After joining i and j, the joined node k is derived, as is shown in Fig. 2(b). The weight w=3 assigned to arc (i,j) expresses the size of the relation R_k. Labeling (b,c,a) in R_i does not perticipate in the resulting join R_k, because its common part (c,a) does not appear in any of tuples, labelings, of R_j. On the othrer hand, R_j does not contain such odd labelings. When both R_i and R_j don't contain any invalid odd labelings at all, the arc (i,j) is called arc-consistent[7,8,11]. And a constraint network is said to be relaxed if all of the arcs are arc-consistent.

(a)



(b)

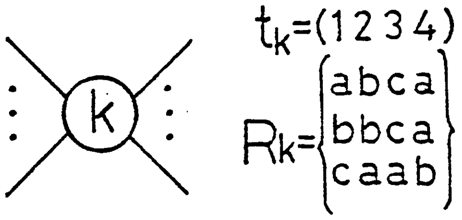


Fig.2 An example of join(i,j) operation.

Repeating the join operation one after another, the constraint network becomes only a single node, with which the final constraint label relations containing all of the answers (i.e. consistent labelings) is associated.

2.4 The Constraint-Propagation Operation

For the efficiency of each join-operation and to keep the working space small, we introduce some operations which are used to make a constraint network relaxed or quasi-relaxed. First we define two operations used to eliminate the odd labelings.

The constraint-propagation from node i to node j, written as prop(i,j), is the operation that eliminates all the odd labelings in R_j that will not participate in the joined label relation. Thus, after performing prop(i,j), R_j is reduced to the following relation:

$$\{r \mid r \in R_j \wedge \exists s (\in R_i)(s[d]=r[d])\}.$$

Note that the prop operation is non-commutative. In fact, prop(i,j) reduces R_j, while prop(j,i) reduces R_i. The mutual-constraint-propagation between two nodes i and j, written as mutual-prop(i,j), is the operation that performes both prop(i,j) and prop(j,i) at once. Apparently, mutual-prop(i,j) is a commutative operation that makes the arc (i,j) arc-consistent. Both prop and mutual-prop include the calculation of the weight w((i,j)) also.

Next, we introduce three kinds of operations that relax the consistent network. The global relaxation is the procedure that makes the constraint network completely relaxed. Its basic mechanism is found in [11], though we implemented so that the mutual-prop is used as many times as possible. The mutual-prop(i,j) inevitably contains a natural join operation used in relational database systems[12] of R_i and R_j on the common units $d(=\bar{t}_i \cap \bar{t}_j)$. To perform the join operation effectively, we prepared two hash tables[13] with the same size, one of which is used for storing R_i and the other for R_j. If an entry of a table contains an element (i.e. a labeling) while the corresponding entry of the other table with the same address is empty, then the element is easily concluded to be an odd labeling. This technique of implementation is the same to the one adopted in the database machine LEECH[14], which enables the mutual-prop operation to be performed as fast as the usual prop operation. Therefore, when two prop operations whose directions are opposite according to the same arc are required, a mutual-prop operation can be used to get rapidly the same result attained by those two prop operations.

The other procedures using mutual-prop operations are the <u>partial relaxation</u> and the <u>local relaxation</u>. The former applies a mutual-prop operation once per every arc throughout the network. In the latter procedure, mutual-prop operations are performed only on the arcs connected to a proper node concerned. The results of both procedures are slightly affected by the order of the arcs on which mutual-props are performed. The local relaxation applied according to node 1 in Fig. 1 eliminates odd labelings (b,e) and (e,b) from R_1. Further, if arc (1,4) is processed earlier than arc (1,2), odd labelings (b,b,a) and (b,d,c) in R_2 are also deleted. Fig. 3(a) shows the result of the global relaxation applied to Fig. 1.

Before going into the details of the algorithm, we clarify some properties of constraint networks. A node which is adjacent both with nodes i and j is said to be bi-connected to i and j.

Proposition 1.

Let i and j be adjacent nodes in a relaxed constraint network. Then, the arc-consistency is maintained after joining nodes i and j except the arcs that connect the joined node to the nodes which were bi-connected to nodes i and j.

Proposition 2.

Let i,j and k be adjacent nodes in a constraint network G. Let G' be the same network as G except that the arc (j,k) is not included. If $\bar{t}_j \wedge \bar{t}_k \leq \bar{t}_i$, then $C_G = C_{G'}$, where C_G and $C_{G'}$ are the sets of all consistent labelings of the network G and G', respectively.

3. The Constraint Synthesizing Algorithm

In the previous section, we introduced three relaxation procedures using constraint propagation operations: the global relaxation, the partial relaxation and the local relaxation. The general framework of constraint synthesizing algorithm is described as follows:

```
1. procedure Consistent_Labeling;
2.     relaxation(initializing_type);
3.     repeat
4.         find the arc with minimum weight and apply join
           operation;
           if the result (i.e. the joined label relation) is
           empty, then stop(no result found);
5.         relaxation(repeating_type)
6.     until the number of arcs becomes zero.
```

It should be noticed that the procedure works correctly even when the 2nd and 5th steps are removed. Step 2 initially relaxes the given constraint network to eliminate odd labelings which apparently does not participate in any of the resulting join. After performing step 4, it is no more guaranteed that the network is relaxed. Step 5, however, incurs some kind of relaxation procedure to recover the relaxedness of the network. Anyway, step 2 and 5 are used to reduce the combinatorial explosion effect by removing odd labelings from some label constraint relations, and to update the weight of each arc.

The loop of 4 and 5 is repeated until all the arcs are joined and deleted or until a joined label relation is found to be empty. In the former case, the final label constraint relation gives all of the solutions, or consistent labelings. In the latter case, it is immediately concluded that there is no solutions. When the constraint network is not connected, each component reduces to a single node. Then, all of the solutions are given by the Cartesian product of the label constraint relations associated to those final single nodes.

Table 1 shows three kinds of important combinations of relaxation procedures being assigned to steps 2 and 5. Method 1 does not practically contain any heuristics. It performs only a weak relaxation once at the beginning in step 2. On the other hand, method 3 always keeps the constraint network completely relaxed. The arc on which the join operation in step 4 should be applied is heuristically chosen by checking the weight of each arc. Thus join operations in method 3 are carried out more rapidly using less working area than that in method 1. But in method 3, it takes much time to accomplish global relaxation itself. Method 2 is an intermediate method between methods 1 and 3. Proposition 1 in Section 2.4 suggests that keeping the network relaxed is sufficiently attained by applying a local relaxation with respect to the arc on which the last join operation has been performed.

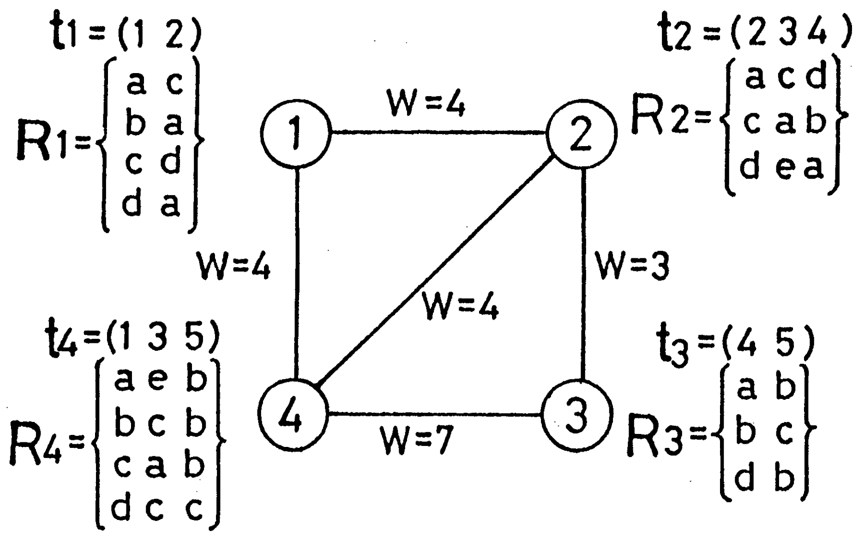
Fig. 3 shows the transition of the constraint network given in Fig. 1 when the method 3 is applied.

4. Experiments

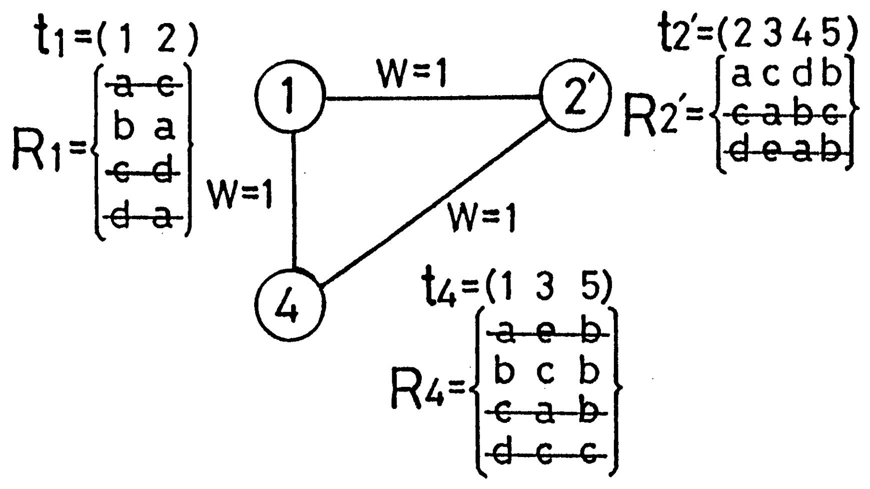
Implementing three methods proposed in Section 3, we try to make clear their efficiency and characteristics. Fig. 4 shows two types of graph structures, N1 and N2, used to produce concrete examples of CLPs. We generated 100 concrete CLPs per each graph

Table 1. Three main combinations of relaxation procedures.

	relaxation (initializing type)	relaxation (repeating type)
method 1	partial relaxation	none
method 2	partial relaxation	local relaxation
method 3	global relaxation	global relaxation



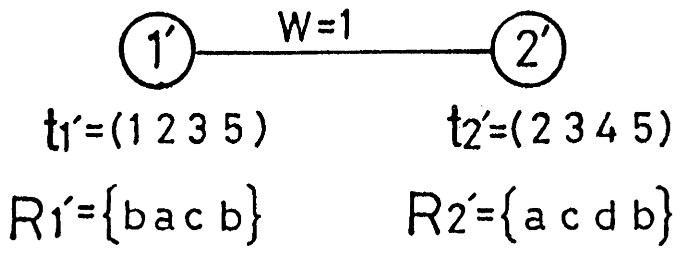
(a) The relaxed constraint network of Fig.1.



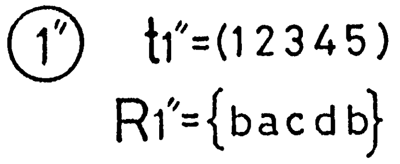
t1=(1 2)
R1={~~a c~~, b a, ~~c d~~, ~~d a~~}
t2'=(2 3 4 5)
R2'={a c d b, ~~c a b c~~, ~~d e a b~~}
W=1
W=1
W=1
t4=(1 3 5)
R4={~~a e b~~, b c b, ~~c a b~~, ~~d c c~~}
1
2'
4

(b) The network after performing join(2,3) of (a), where tuples marked with ——— are removed by the following relaxation(repeating type).

Fig.3 Transition of the sample network during the progress of the method 3. (continued)



(c) The network after performing join(1,4) of (b), which is already relaxed.



(d) The result derived after join(1',2').

structure by using pseudo random numbers. As for N1, the number of labels is 8 and the total number of labelings in all label constraint relations, $R_1, \ldots, R_8$, is 240 (i.e. 30 labelings in average per relation). As for N2, the number of labels is 10 and the total number of labelings is 320 (i.e. 40 labelings in average per relation). To generate each concrete CLP, uniform pseudo random numbers are used not only to make new labelings one after another but also to determine to which node the next labelings is to be added.

The more the number of labels becomes, the more the ratio of odd labelings increases. The average degree of N1 is 3.75 which is greater than that of N2, 2.5. As the average degree increases, the constraint becomes more tight; thus, the number of solutions tends to be less. To sum up, the conditions of a constraint network that make the number of final solutions small and the computation time short are as follows:

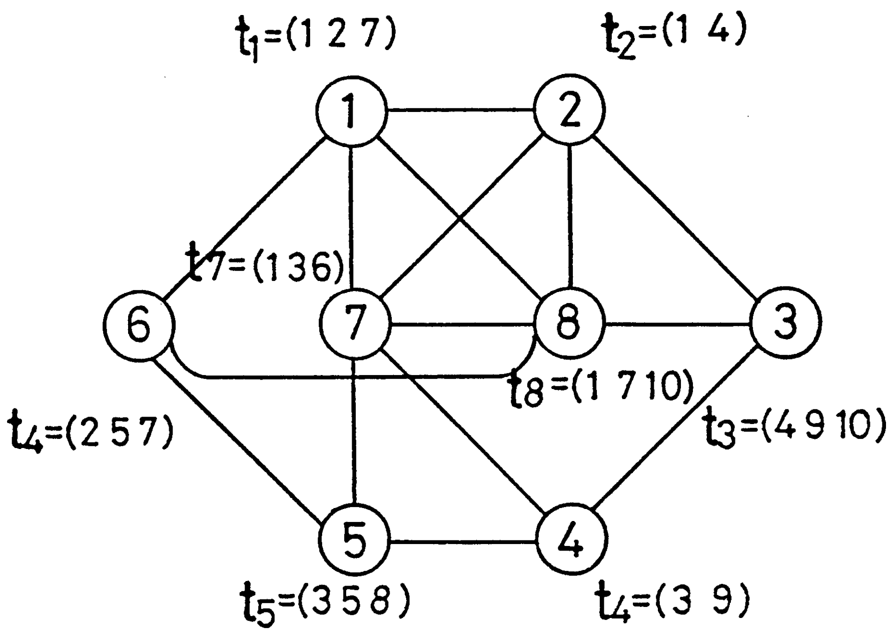
i) the average of degrees is large,

ii) the number of labels are large, and

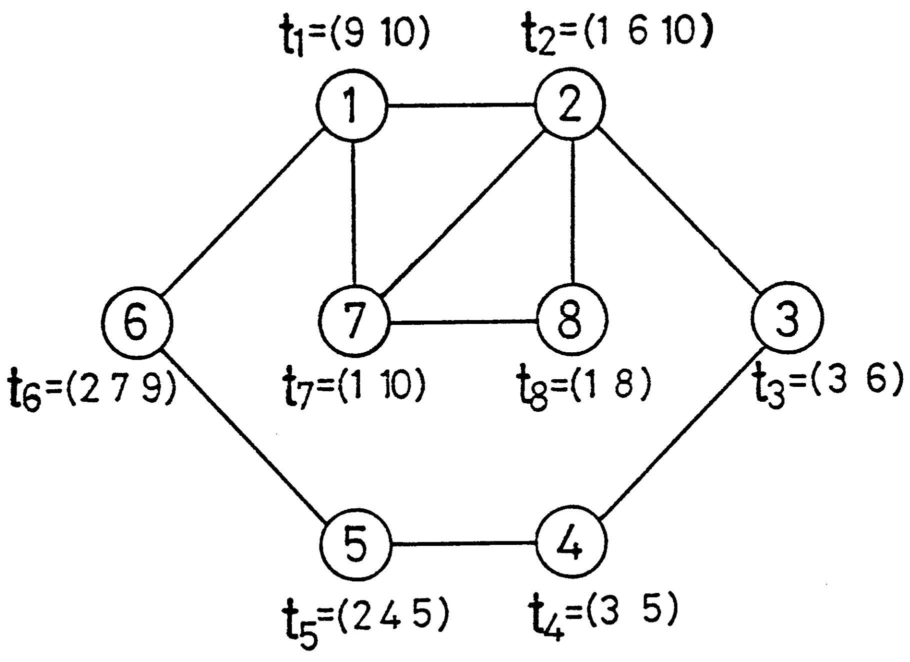
iii) the average size of the label constraint relations is small.

Actually, the experiment shows the average number of solutions of the cases N1 and N2 are 206.3 and 3100.1, respectively.

The generated 100 concrete CLPs per each graph, N1 and N2, are processed by the three methods shown in Table 1. The average CPU time to solve a CLP is shown in Table 2. As we mentioned in Section 3, method 2 gives the best results: the moderate relaxation procedure is effective (comparing with method 1), but



(a) N1 structure.



(b) N2 structure.

Fig.4 The graph structures used for the experiments.

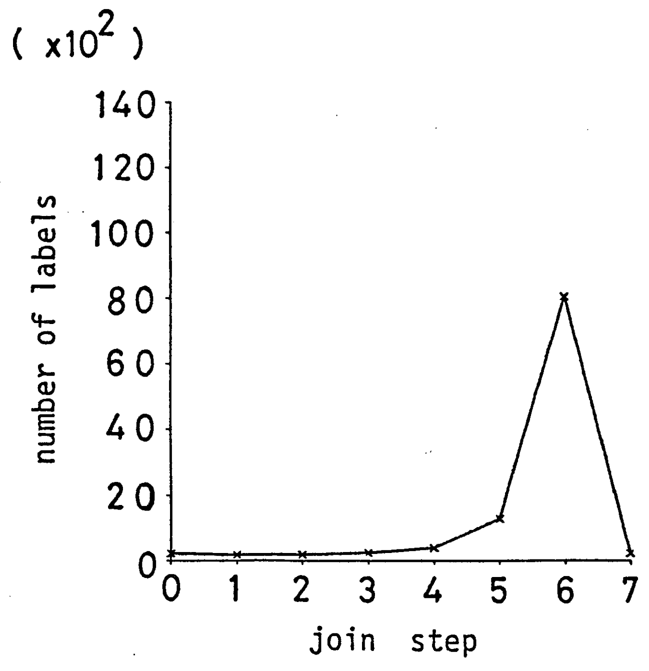
Table 2. Average CPU time per CLP (sec).

	method 1	method 2	method 3
N1	2.391	2.244	3.173
N2	4.982	3.766	3.973

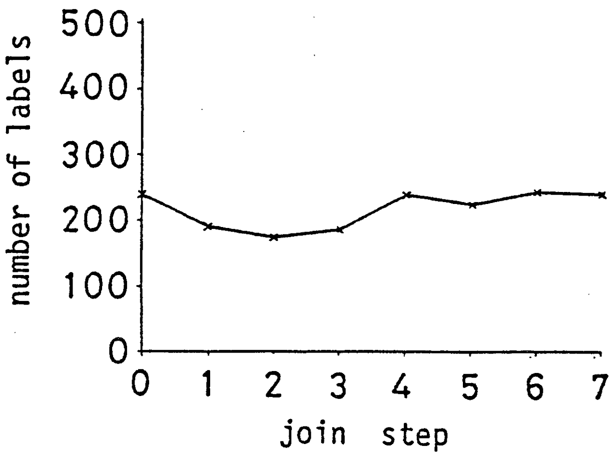
there is no need of full relaxation (comparing with method 3). In case of N1, we measured the variances of CPU time of methods 1, 2 and 3, which are 2.653, 0.214 and 0.210, respectively. The result shows that, comparing with method 1, the other two methods are very stable in computation time. And, the method 2 is actually as stable as method 3. This fact is proved also by measuring the correlation of CPU time of each CLP. The correlation coefficient of methods 1 and 3 is 0.368, while that of 2 and 3 is 0.873, which shows high correlation.

Next, let us pay attention to the working space. In the case of method 1, about 10% of 100 problems of N1 require extraordinarily long computation time, as for one of a typical case of which the transition of the size of working space is shown in Fig. 5. The total number of all the surviving labelings after the execution of each join step is expressed by the vertical axis; thus, it also gives the size of the working space. As a matter of course, the final value of the vertical axis, or the number of solutions, is always the same, 240, in each case of (a), (b) and (c) in Fig. 5. In the case of method 1 (Fig. 5(a)), a combinatorial explosion is observed at step 5 and 6, where more than 8,000 labelings survive. On the other hand, in the cases of methods 2 and 3, which are shown in Fig. 5(b) and (c), respectively, the number of remaining labelings is always suppressed under 300. Thus, method 1 is inappropriate especially when the working space is limitted.

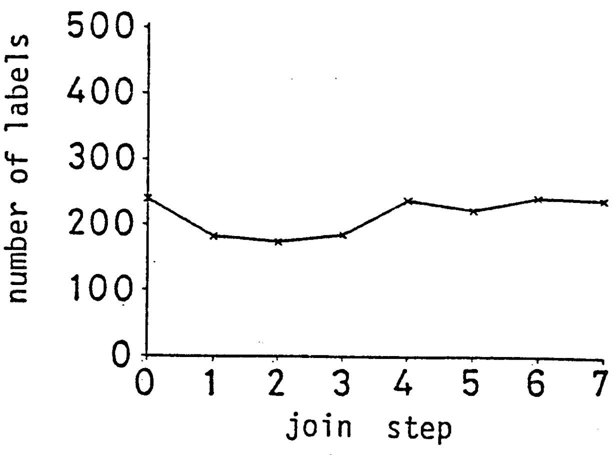
As discussed above, method 2 gives the best result in terms both of time and space efficiency. In [4], several depth-first



(a) Method 1.



(b) Method 2.



(c) Method 3.

Fig.5 Transition of the size of work area.

algorithms are compared experimentally by using N queens problem. According to the results proved in [4], we can find that the relative efficiency of standard backtracking, forward-checking and looking-ahead are very similar to that of our methods 1, 2 and 3, respectively. It means that forward-checking, which is intermediate method between standard backtracking and looking-ahead, attains moderate heuristics giving best trade-offs.

Our experimental programs are written in Fortran 77 and run on Perkin-Elmer 3220. We made use of hashing technique (chaining) to implement the join operation and the constraint propagation procedure, as was referred to in Section 2.4.

5. Conclusion

A CLP is defined by a set of units, U, a set of labels, L, a unit constraint relation, T, and a label constraint relation, R. In the foregoings, we first extended the conventional definition of CLP so that unit relations with different dimensionalities may coexist. We introrduced constraint networks showing any CLP can equivalently be expressed by a constraint network. Then, a novel constraint synthesizing algorithm which reduces the given constraint network to a single node containing all of the final consistent labelings was proposed. We defined three types of concrete algorithms and proved their efficiency and characteristics experimentally.

CLP is a kind of search problem in artificial intelligence, where it must be noted that too much stress should not be laid on finding the optimal path only. Our experimental result also shows that a moderate heuristics gives the best result.

Since the termination criterion of the algorithm is that there is no more arcs in the network, it works correctly even when the given constraint network is not connected. Further, whenever it is found that every label relation contains only one labeling (,such a case is seen in Fig. 3(b)), the algorithm can immediately be terminated, giving one solution. The algorithm is suitable to parallel processing, because the join operation and the constraint propagation are locally executable as for an arc concerned. If we have sufficient number of processors, the computation time is reduced to $O(\log|V|)$.

Acknowledgment. We would like to thank Professors R. M. Haralick and L. G. Shapiro of Virginia Tech (presently with Machine Vision International) who introduced the consistent labeling problem to the first author.

References

[1] Rosenfeld,A., Hummel,R.A. and Zucker,S.W.: Scene labeling by relaxation operations, IEEE Trans. Syst. Man & Cybern.,SMC-6,6(1976),420-433.

[2] Waltz,D.L.: Understanding line drawings of scenes with shadows, in The Psychology of Computer Vision, ed. P.H.Winston, McGraw-Hill,NY(1975).

[3] Ullmann,J.R.: An algorithm for subgraph isomorphism, J.ACM, 23,1(1976),31-42.

[4] Haralick,R.M., and Elliott,G.L.: Increasing tree search efficiency for constraint satisfaction problem, Artif. Intell., 14,3(1980),263-313.

[5] Haralick,R.M. and Shapiro,L.G.: The consistent labeling problem, Part I, IEEE Trans. Pattern Anal. Machine Intell., PAMI-1,2(1979),173-184, and Part II, ibid.,PAMI-2,3(1980),193-203.

[6] Nudel,B.: Consistent-labeling problems and their algorithms: expected-complexities and theory-based heuristics, Artif. Intell., 21,1&2(1983),135-178.

[7] McGregor,J.J.: Relational consistency algorithms and their application in finding subgraph and graph isomorphisms, Inf. Sci.,19,3(1979),229-250.

[8] Mackworth,A.K.: Consistency in networks of relations, Artif. Intell.,8,1(1977),99-118.

[9] Gaschnig,J.: A general backtrack algorithm that eliminates most redundant tests, Proc. Int. Conf. on Artif. Intell., (1977),457.

[10] Montanari,U.: Networks of constraints: fundamental properties and applications to picture processing, Inf. Sci.,7,2 (1974),95-132.

[11] Freuder,E.C.: Synthesizing constraint expressions, C.ACM,21, 11(1978),958-966.

[12] Date,C.J.: An Introduction to Database Systems (3rd ed.), Addison-Wesley, Reading,Mass.(1981).

[13] Nishihara,S.: Hashing techniques and their applications, J. Inf. Process. Society Japan,21,9(1980),980-991.

[14] McGregor,D.R., Thomson,R.G. and Dawson,W.N.: High performance hardware for database systems, Proc. 2nd Intnat'l Conf. VLDB, North-Holland,(1976),103-116.

INSTITUTE OF INFORMATION SCIENCES AND ELECTRONICS
UNIVERSITY OF TSUKUBA
SAKURA-MURA, NIIHARI-GUN, IBARAKI 305 JAPAN

REPORT DOCUMENTATION PAGE	REPORT NUMBER ISE-TR-85-49

TITLE

Consistent Labeling Methods Using Constraint Networks

AUTHOR(S)

Seiichi Nishihara
Katsuo Ikeda

Inst. of Information Sciences and Electronics
University of Tsukuba

REPORT DATE February 25, 1985	NUMBER OF PAGES 25
MAIN CATEGORY Problem Solving and Search	CR CATEGORIES F.2.2,I.2.8,I.2.10,I.4.8

KEY WORDS
backtracking, combinatorial algorithm, inference, filtering, search, pattern analysis, consistent labeling, relaxation, constraint propagation, scene labeling, constraint network

ABSTRACT

Problems of finding totally consistent interpretations when an object cinsisting of many parts and their locally legal interpretations are given are found in some different areas such as image understanding and artificial intelligence. A novel breadth-first algorithm is proposed in this paper. First, it is shown that any consistent labeling problem can equivalently be expressed by a constraint network. Then, a general structure of the procedure, which repeats relaxing the network and joining two nodes in the network, is proposed. Finally, we introduce three main types of concrete algorithms with different levels of heuristics and compare their efficiency and characteristics by computer simulation.

SUPPLEMENTARY NOTES