ISE-TR-83-42

CONSTRAINT SYNTHESIZING BY NETWORK REDUCTION FOR THE CONSISTENT LABELING PROBLEM

by

Seiichi Nishihara

Katsuo Ikeda

November 30, 1983

INSTITUTE
OF
INFORMATION SCIENCES AND ELECTRONICS

UNIVERSITY OF TSUKUBA

Constraint Synthesizing by Network Reduction for the Consistent Labeling Problem

Seiichi NISHIHARA and Katsuo IKEDA

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

Abstract

There are two principal strategies for the consistent labeling problem (CLP): the depth-first approach and the breadth-first approach. The former is typified by backtracking, which often causes a costly thrashing phenomenon. This paper proposes a novel algorithm that takes the breadth-first approach. First, it is shown that every CLP can be equivalently expressed as a constraint network. Then, an efficient algorithm, which repeats relaxing the network and joining two nodes in the network, is proposed. The algorithm straightforwardly reduces the given network to a single node containing all of the final consistent labelings.

1. Introduction

The purpose of solving the consistent labeling problem (CLP) is to provide an efficient way of analyzing or understanding an object whose structural description is given by its primitive parts and their interrelationships. Application fields for the CLP extend from image understanding problem, such as scene labeling, to puzzles, such as the N-queens problem. There are two principal strategies for solving the CLP: the depth-first approach and the breadth-first approach[1]. One is a trial-and-error method, which inherently induces backtracking. To reduce the costly thrashing phenomenon caused by the simple brute-force method of backtracking, many refinements of the depth-first approach have been proposed, among which forward checking proved most efficient[2]. The other approach is called by many terms, such as filtering, constraint-propagation, constraint-synthesizing or relaxation of a constraint network[3,4].

First, we give the definition of the CLP, which is a generalized version of the one given by Haralick and Shapiro[5]. Let $U=\{1, \ldots ,M\}$ be a set of units and let L be a set of labels. Units represent the primitive parts of an object to be analyzed or understood. Labels are possible interpretations of units. Then a CLP is represented by a compatibility model (U,L,T,R), where $T \subseteq \bigcup_{1 \le i \le M} U^i$ is the set of tuples of units which mutually constrain one another and $R=\{R_t \subseteq L^{|t|} \mid t \in T\}$ is the set of constraint relations. Here, $|t|$, called the dimension, is the number of components in tuple t. Our problem is to find all consistent labelings, or M-tuples of labels $(l_1, \ldots ,l_M)$ assigned to units $(1, \ldots ,M)$, satisfying the constraints T and R simultaneously.

Freuder[3] has developed a constraint synthesizing algorithm that incrementally updates the constraint network by adding new nodes corresponding to constraints of higher order. At first, the constraint network is merely a set of isolated nodes with each of which a unit and its unary constraint of labels are associated. One node is inevitably added to the

network for every combination of units, and this may cause algorithm inefficiency. As is shown in Sec. 3, however, there is no need to introduce as many nodes as 2^M-1 but only the same number of nodes as the number of constraint relations.

2. Constraint Networks

2.1 Definitions

A constraint network is an undirected graph (V,E) with a weighting function w which assigns an integer to each arc. With each node in V, a tuple of units and a constraint relation (a set of possible labelings for the tuple) are associated. For any two nodes 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 associated with the two nodes. Let R_i and R_j be the constraint relations which put restraint upon tuples t_i and t_j, respectively. We now define the *join* operation, which is repeatedly applied to the constraint network until it reduces to a single node. Our definition of join is essentially the same as that of the natural join of the relational data model[6]. Let d be the set of units common to both $\overline{t_i}$ and $\overline{t_j}$, which are the sets of units in tuples t_i and t_j, respectively. Then the join of R_i and R_j over d is a relation in which each tuple consists of a tuple from R_i concatenated with a tuple from R_j that contains the same d-labels. In a constraint network, the join operation also causes a merge of nodes i and j into one node, deleting the arc (i,j). Every arc connecting i or j to another node k remains as an arc connecting the joined node and node k. When neither R_i nor R_j contains a tuple that does not participate in the join, the arc (i,j) is said to be arc-consistent. The weighting function w assigns each arc (i,j) a *weight* that is the size of the joined node that will be obtained after joining R_i and R_j. The weighting function w is used to determine heuristically on which arc the next join operation should be performed.

Fig. 1 shows an example of a part of a constraint network. The common-unit set d is $\{2, 3\}$. As shown in Fig.1, the size of the join of R_i and R_j on d is three, which, from the definition, becomes the weight of arc (i,j). Tuple (b,e,d) in R_i does not participate in the resulting join R_k, because its common part (e,d) does not appear in any of the tuples of R_j. In R_j, (a,e,g) is such an odd tuple. A constraint network is said to be relaxed if all of the arcs are arc-consistent.

We now define two operations used to relax a constraint network. The constraint propagation from node i to node j, written as *prop*(i,j), is the operation that eliminates all the odd tuples in R_j that will not participate in the joined relation R_k. Note that the prop operation is non-commutative. In fact, in Fig. 1, prop(i,j) deletes tuple (a,e,g) from R_j, while prop(j,i) deletes tuple (b,e,d) from R_i. The mutual constraint-propagation between two nodes i and j, written as *mutual-prop*(i,j), gives the same result that both prop(i,j) and prop(j,i) attain. Apparently, mutual-prop operation is commutative. It is easy to see that any compatibility model (U,L,T,R) defined in Sec.1 can be expressed by an equivalent constraint network. Fig. 2 is an example of CLP represented as a constraint network. A filtering method that synthesizes the given constraints to get all of the consistent labelings for the units $\{1, \ldots, M\}$ is described as follows:

```
procedure Constraint-Synthesize;
    repeat
        relaxation-phase: relax the constraint network;
        join-phase: select an arc and join
    until (number of arcs=0) or (join= φ ).
```

It should be noticed that this procedure works correctly even when the relaxation-phase is removed, that is, the relaxation-phase is introduced only for efficiency's sake. Before going into the details of the algorithm, we clarify some properties of constraint networks. When

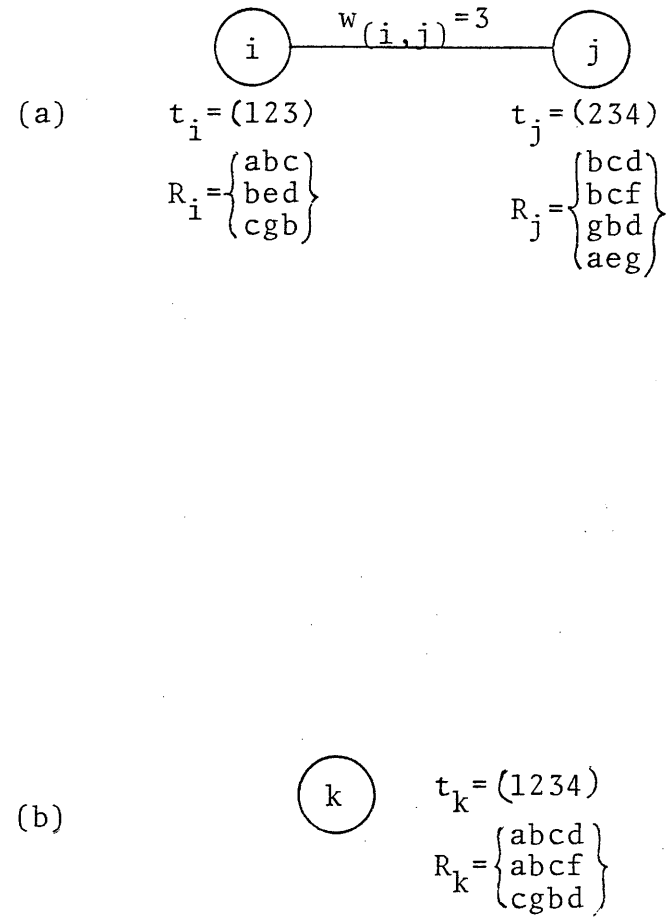

Fig.1 Example of (a) a part of a constraint network, and (b) the obtained join of R_i and R_j.

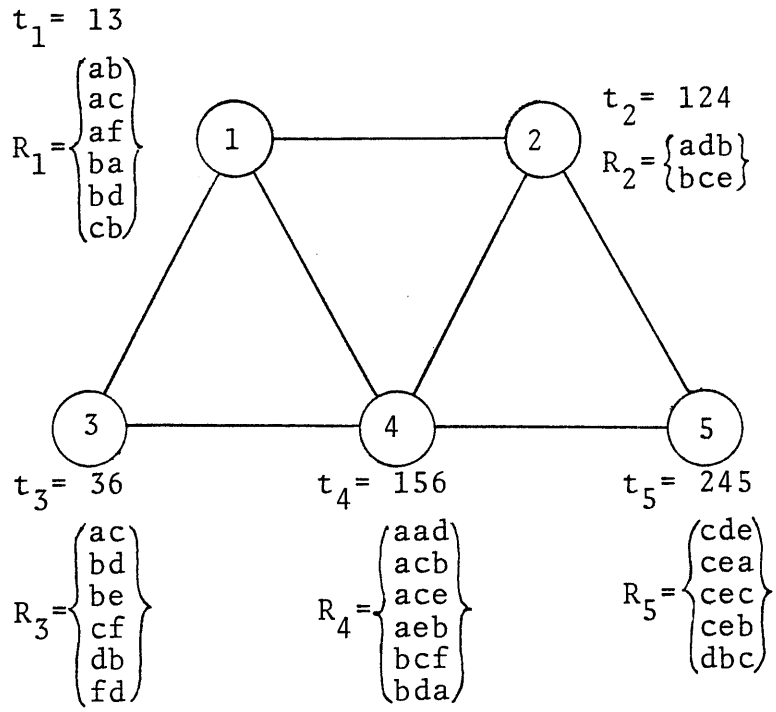

Fig. 2 A simple constraint network.

node k is adjacent both with nodes i and j, node k 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 only arcs whose arc-consistency is no longer guaranteed after joining nodes i and j are those that connect the joined node to 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 $\overline{t}_j \cap \overline{t}_k \subseteq \overline{t}_i$, then $C_G = C_{G'}$, where C_G and $C_{G'}$ are the sets of all consistent labelings.

Corollary 1:

If $\overline{t}_j \subseteq \overline{t}_i$ or $\overline{t}_k \subseteq \overline{t}_i$, then $C_G = C_{G'}$.

Corollary 2:

Let $\overline{t}_j \cap \overline{t}_k \subseteq \overline{t}_i$ and arcs (i,k) and (j,k) be arc-consistent. Then the arc-consistency of arc (i,k) is maintained if R_i is not changed (i.e. not reduced) by the prop(j,i) operation.

3. The Algorithm

The detailed algorithm of the Constraint-Synthesize procedure given in Sec. 2 is described here. Fig. 3 shows the main program, where lines 6-15 and lines 16-17 correspond to the relaxation-phase and join-phase in the procedure, respectively. While the main program runs correctly even when lines 2-4 and 6-15 are omitted, this portion of the program reduces the combinatorial explosion effect by removing inconsistent tuples of

labels which will not contribute to a final consistent labeling. Q and P are used to hold pairs (i.e. arcs) and ordered pairs of nodes for which it is worth applying the MUTUAL-PROP procedure (Fig.4(a)) and PROP procedure (Fig.4(b)), respectively. We implemented the algorithm in Fortran 77, in which the operations mutual-prop, prop and join are effectively performed by using hashing techniques similar to the rough selection mechanism of the LEECH machine[7].

A concrete example showing how a given constraint network is processed by the algorithm is illustrated in Fig. 5, where the constraint network of Fig. 2 is processed. The only final consistent labeling found is (a,d,b,b,c,e).

```
1   begin
2      Q ←{(i,j)|(i,j)εE, i<j};
3      w(i,j) ←∞ for each arc (εE);
4      P ←φ;
5      repeat
6         while Q≠φ do
7            begin
8               select and delete an arc (i,j) from Q;
9               MUTUAL-PROP(i,j)
10           end;
11        while P≠φ do
12           begin
13              select and delete a pair <i,j> from P;
14              PROP(i,j)
15           end;
16        select the arc (i,j) with minimum weight;
17        JOIN(i,j)
18     until #arcs=0
19  end.
```

Fig. 3 The main program.

```
20  procedure MUTUAL-PROP(i,j);
21     begin
22        if t̄_i ⊆ t̄_j or t̄_j ⊆ t̄_i then JOIN(i,j);
23        mutual-prop(i,j);
24        calculate and update the arc-weight of (i,j);
25        if the arc-weight=0 then terminate(not found);
26        if R_i is changed(reduced) then
27           P← P∪{<i,k>|(i,k)εE-Q, k≠j};
28        if R_j is changed(reduced) then
29           P← P∪{<j,k>|(j,k)εE-Q, k≠i}
30     end.
```

Fig. 4(a) MUTUAL-PROP procedure.

```
31  procedure PROP(i,j);
32     begin
33        prop(i,j);
34        calculate and update the arc-weight of (i,j);
35        if R_j is changed(reduced) then
36           P← P∪{<j,k>|(j,k)εE, k≠i}
37     end.
```

Fig. 4(b) PROP procedure.

```
38  procedure JOIN(i,j);
39     begin
40        R_i ← join(R_i,R_j);
41        Q ← Q∪{(i,k)|(i,k),(j,k)εE, k≠i,j};
42        P ← P∪{<i,k>|[(i,k) E,k≠j] or [(j,k) E,k≠i]};
43        E ← [E∪{(i,k)|(j,k)εE}]-{(j,k)|(j,k)εE}
44     end.
```

Fig. 4(c) JOIN procedure.

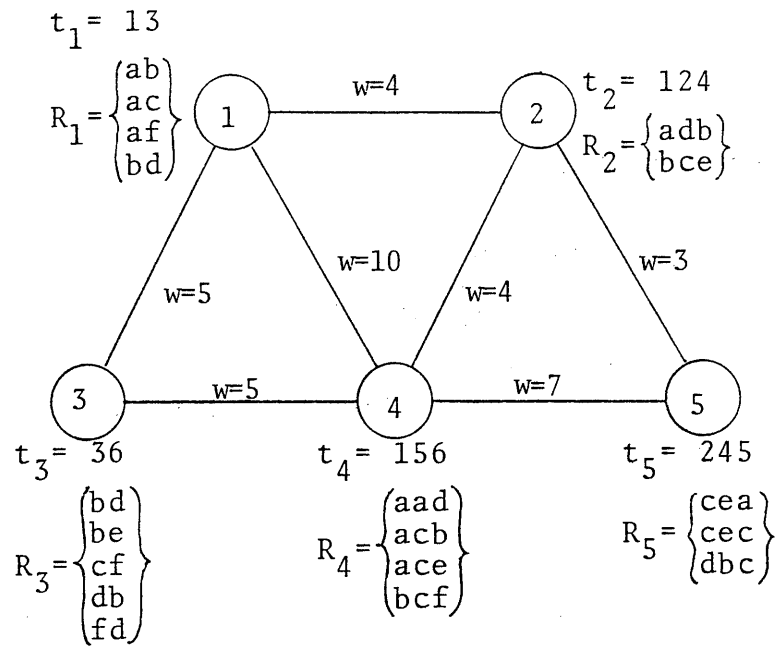

Fig. 5(a) The relaxed network of Fig.2.

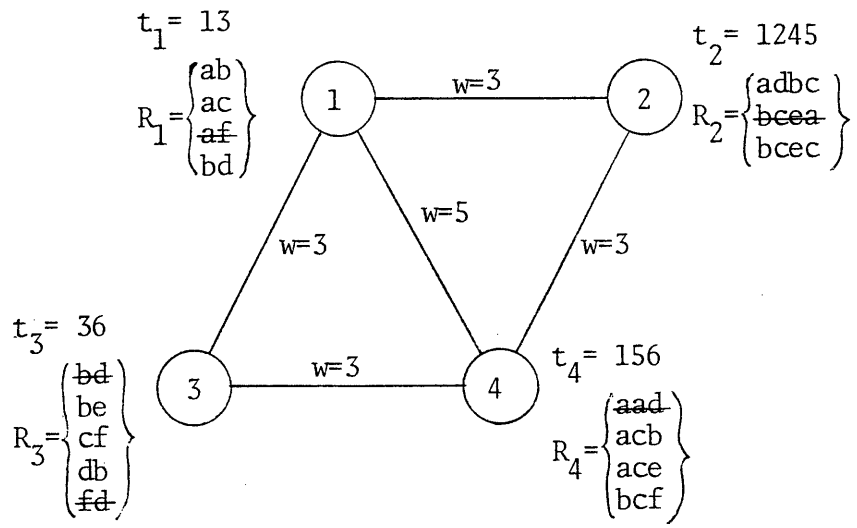

Fig. 5(b) The network after performing join(2,5) of (a), where tuples marked with —— are removed by the following relaxation-phase.

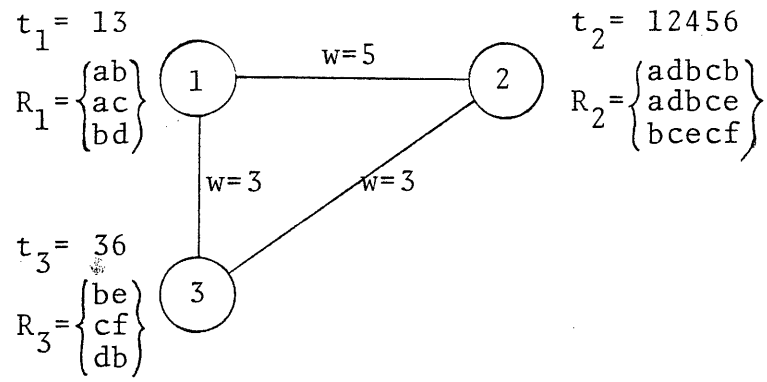

Fig. 5(c) The network after performing join(2,4) of (b), which is already relaxed.

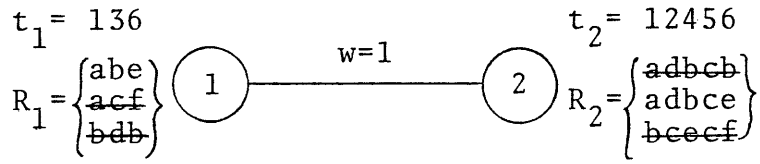

Fig. 5(d) The network after performing join(1,3) of (c), where tuples marked with ——— are removed by the following relaxation.

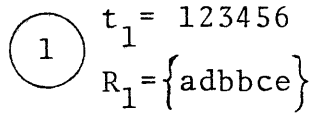

Fig. 5(e) The result derived after join(1,2).

Fig. 5 Transition of the sample network during the progress of the algorithm.

4. Conclusion

A new filtering algorithm for the consistent labeling problem has been proposed that makes full use of the relational join operations. In the algorithm, a heuristic is used to determine the arc with which the next join operation should be performed. This achieves rapid join operations and keeps the working space small. The given constraint network defined in this paper is straightforwardly reduced by the algorithm until it becomes only a single node. The constraint network does not contain any node that has no corresponding constraint relation, so it is kept very small as compared with that of Freuder's[3].

Further, since the termination criteria of the algorithm is that there is no arc in the network, it works correctly even when the given constraint network is not connected. And the dimensions of tuples in T need not be the same. Our filtering algorithm is especially suited to the case that the number of candidate labels for each unit is comparatively large, and is also suitable for parallel processing.

Acknowledgments

The authors would like to thank Professors R. M. Haralick and L. G. Shapiro of Virginia Tech for introducing the first author to the consistent labeling problem. The authors also thank Professor J. W. Higgins of University of Tsukuba for his valuable suggestions.

References

1. Nudel,B.: Consistent-labeling problems and their algorithms: Expected-complexities and theory-based heuristics, Artif. Intell., 21(1983),135-178.

2. Haralick,R.M., Elliott,G.L.: Increasing tree search efficiency for constraint satisfaction problems, Artif. Intell., 14(1980),263-313.

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

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

5. Haralick,R.M., Shapiro,L.G.: The consistent labeling problem, Part I, IEEE Tr. PAMI, 1,2(1979),173-184.

6. Date,C.J.: An Introduction to Database Systems, 2nd ed., Addison-Wesley(1977).

7. McGregor,D.R., Thomson,R.G., Dawson,W.N.: High performance hardware for database systems, Proc. 2nd Internatl. 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-83-42

TITLE

Constraint Synthesizing by Network Reduction for the Consistent Labeling Problem

AUTHOR(S)

Seiichi Nishihara

Katsuo Ikeda

Inst. of Information Sciences and Electronics

University of Tsukuba

REPORT DATE	NUMBER OF PAGES
November 30, 1983	15
MAIN CATEGORY: Problem Solving and Search	CR CATEGORIES: F.2.2,I.2.8,I.2.10,I.4.8,I.5.0

KEY WORDS

backtracking, combinatorial algorithm, consistent labeling, filtering, constraint network, constraint propagation, constraint synthesizing, network consistency, join, relaxation, scene labeling, search

ABSTRACT

There are two principal strategies for the consistent labeling problem (CLP): the depth-first approach and the breadth-first approach. The former is typified by backtracking, which often causes a costly thrashing phenomenon. This paper proposes a novel algorithm that takes the breadth-first approach. First, it is shown that every CLP can be equivalently expressed as a constraint network. Then, an efficient algorithm, which repeats relaxing the network and joining two nodes in the network, is proposed. The algorithm straightforwardly reduces the given network to a single node containing all of the final consistent labelings.

SUPPLEMENTARY NOTES