An Optimal $k$-Consistency Algorithm

Martin C. Cooper
Département d'Informatique, Université de Bordeaux I, 33405 Talence, France

ABSTRACT
This paper generalizes the arc-consistency algorithm of Mohr and Henderson [4] and the path-consistency algorithm of Han and Lee [2] to a $k$-consistency algorithm (arc-consistency and path-consistency being 2-consistency and 3-consistency, respectively). The algorithm is a development of Freuder's synthesis algorithm [1]. It simultaneously establishes $i$-consistency for each $1 \leq i \leq k$. It has worst-case time and space complexity which is optimal when $k$ is a constant and almost optimal for all other values of $k$.

In the case that all order-$i$ constraints exist for all $1 \leq i \leq n$, this algorithm is a solution to the consistent labeling problem with almost optimal worst-case time and space complexity.

1. Introduction

The consistent labeling problem is a general problem statement encompassing many well-known and computationally difficult problems, such as graph coloring, school timetabling and image labeling. The problem statement consists of a set of nodes $N = (1, 2, \ldots, n)$, a set of labels $A$ and a set of constraints $C$. A consistent labeling $(l_1, l_2, \ldots, l_n)$ is an assignment of labels $l_i \in A$ to nodes $i \in N$ so that the constraints $C$ are satisfied. A $k$-ary constraint $C_k(n_1, n_2, \ldots, n_k)$ is a list of permitted labelings for the nodes $n_1, n_2, \ldots, n_k$.

For example, in the graph coloring problem, there are only binary ($k = 2$) constraints and $C_2(n_1, n_2)$ is the set of all pairs of colors $(a, b)$ such that $a \neq b$, for all pairs of adjacent nodes $n_1, n_2$.

A complete solution to the consistent labeling problem is a listing of all consistent labelings, which requires at least $O(na^n)$ time and space in the worst case, where $n = |N|$ and $a = |A|$. To mitigate the combinatorial explosion, the concepts of arc-consistency and path-consistency were introduced [3, 5].
Freuder [1] generalized these concepts to k-consistency (arc-consistency and path-consistency being 2-consistency and 3-consistency, respectively). The idea is that many inconsistent combinations of k − 1 labels are eliminated quickly by a k-consistency algorithm, in order to produce a combinatorial reduction in the computational resources required by a subsequent exhaustive search. The combinatorial explosion can, however, only be mitigated and not avoided, since in the worst case the output consists of O(a^n) n-tuples.

2. The Algorithm KS

The algorithm KS presented in this section returns a set of labelings R'(N_i) for each cardinality-i subset N_i of N, for each i ≤ k. Each R'(N_i) is initialized with the set of legal labelings C(N_i) for N_i given by the input constraints C. If a labeling is illegal for a set of nodes N_i, this implies that some labelings for subsets or supersets of N_i will also be illegal. Illegal labelings are propagated in R until no more propagation is possible. In this state R is said to be totally relaxed. We show in the following section that, in a totally relaxed state, R is i-consistent for each i ≤ k.

In KS, a cardinality-i subset N_i of N is represented by an i-tuple (n_1, ..., n_i) such that n_1 < n_2 < ⋯ < n_i. If N_i is such an i-tuple and L_i is an i-tuple of labels, then R'[N_i, L_i] = true if and only if L_i ∈ R'(N_i). LIST is the list of labelings (N_i, L_i) which have been found to be illegal but which have not yet been propagated. When a labeling (N_i, L_i) is appended to LIST, M[N_i, L_i] is changed from 0 to 1 to mark (N_i, L_i) so that it will not be added to LIST more than once. LIST is initialized in Step 1 with all labelings (N_i, L_i) such that L_i ∈ C(N_i).

Step 2A propagates an illegal labeling (N_i, L_i) to supersets of N_i of cardinality i + 1. For brevity we have used the following notation: if N_i = (n_1, ..., n_i) and L_i = (l_1, ..., l_i), then N_i ∪ (n') means (n_1, ..., n_i, n', n_{i+1}, ..., n_i) where j is such that n_j < n' < n_{j+1}; L_i ∪ (l') means (l_1, ..., l_j, l', l_{j+1}, ..., l_i) for the same value of j. If the labeling of N_i by L_i is illegal then the labeling of N_i ∪ (n') by L_i ∪ (l') is illegal for all n' ∈ N and for all l' ∈ A.

Step 2B propagates the illegal labeling (N_i, L_i) to cardinality-(i − 1) subsets of N_i. If N_i = (n_1, ..., n_i) and L_i = (l_1, ..., l_i), we use the notation N_i − (n_i) to mean (n_1, ..., n_{i−1}, n_{i+1}, ..., n_i) and L_i− (l_i) to mean (l_1, ..., l_{i−1}, l_{i+1}, ..., l_i). Counter[N_i, L_i, n'] stores the size of the support for the labeling (N_i, L_i) at node n', in other words the number of labels l' ∈ A such that (N_i ∪ (n'), L_i ∪ (l')) ∈ R'^+1. The notion of support was first defined by Mackworth [3]. If Counter[N_i, L_i, n'] falls to 0 for some n' ∈ N, then (N_i, L_i) cannot be extended to a labeling of N_i ∪ (n'), and is therefore an inconsistent labeling.
KS \( k \):

\{Step 1: initialization\}

\[
\text{LIST} := \text{Empty\_list}; \ R := C; \ M := 0; \ \text{Counter} := a;
\]

for \( i := 1 \) to \( k \) do

for each \( i \)-tuple \( N_i \) of nodes \( n_i < \cdots < n_i \) do

for each \( i \)-tuple \( L_i \) of labels \( l_1, \ldots, l_i \) do

if \( R[N_i, L_i] = \text{false} \) then

begin

Append(LIST, \( (N_i, L_i, i) \));

\( M[N_i, L_i] := 1 \);

end;

\{Step 2\}

while LIST \( \neq \text{Empty\_list} \) do

begin

Remove an element \( (N_i, L_i, i) \) from LIST;

\{2A\} if \( i < k \) then \{propagate to level \( i + 1 \)\}

for each \( n' \not\in N_i, l' \in A \) do

begin

\( N_{i+1} := N_i \cup (n') \); \( L_{i+1} := L_i \cup (l') \);

if \( M[N_{i+1}, L_{i+1}] = 0 \) then

begin

Append(LIST, \( (N_{i+1}, L_{i+1}, i + 1) \));

\( M[N_{i+1}, L_{i+1}] := 1 \);

\( R^{i+1}[N_{i+1}, L_{i+1}] := \text{false} \);

end;

end;

\{2B\} if \( i > 1 \) then \{propagate to level \( i - 1 \)\}

for \( j := i \) to \( 1 \) do

begin \{assume \( N_i = (n_1, \ldots, n_i), L_i = (l_1, \ldots, l_i) \}\}

\( N_{i-1} := N_i - (n_j); \ L_{i-1} := L_i - (l_j); \)

\( \text{Counter}[N_{i-1}, L_{i-1}, n_j] := \text{Counter}[N_{i-1}, L_{i-1}, n_j] - 1; \)

if \( \text{Counter}[N_{i-1}, L_{i-1}, n_j] = 0 \) and \( M[N_{i-1}, L_{i-1}] = 0 \) then

begin

Append(LIST, \( (N_{i-1}, L_{i-1}, i - 1) \));

\( M[N_{i-1}, L_{i-1}] := 1 \);

\( R^{i-1}[N_{i-1}, L_{i-1}] := \text{false} \);

end

end
KS is a refinement of Freuder's synthesis algorithm [1]. The important difference is the optimization using Counters, as employed by Mohr and Henderson for arc-consistency [4] and Han and Lee for path-consistency [2]. The other difference is that in Freuder's algorithm the constraints $R_1, \ldots, R'$ are totally relaxed before adding the order-$(i+1)$ constraints, whereas in KS the choice of the order in which to propagate illegal labelings is not fixed.

3. Proof of KS

In this section we prove the correctness of KS. For this we require the following definitions.

The labeling $(N_i, L_i)$ is a sublabeling of $(N_k, L_k)$ if $N_k = (n_1, \ldots, n_k)$, $L_k = (l_1, \ldots, l_k)$, $N_i = (n_{i_1}, \ldots, n_{i_j})$ and $L_i = (l_{i_1}, \ldots, l_{i_j})$ for some $1 \leq i < \cdots < j \leq k$. $(N_k, L_k)$ is an extension of $(N_i, L_i)$ if $(N_i, L_i)$ is a sublabeling of $(N_k, L_k)$. An assignment of labels $L_k = (l_1, \ldots, l_k)$ to nodes $N_k = (n_1, \ldots, n_k)$ k-satisfies the set of constraints $R$ if, for each sublabeling $(N_i, L_i)$ of $(N_k, L_k)$, $L_i \in R'({N_i})$. We can describe $R$ as a set of constraints because of the equivalence between a constraint and a set of labelings.

$$R = \{R'(N_i) : N_i \subseteq N, |N_i| = i, 1 \leq i \leq k\}$$ is a k-solution to the consistent labeling problem $(C, N, A)$ if:

1. $R'(N_i) \subseteq C'(N_i)$ $\forall N_i \subseteq N$ of cardinality $i$, $\forall i$ ($1 \leq i \leq k$).
2. $(L_i \in R'(N_i)) \land (N_i \subseteq N_k \subseteq N) \land (|N_i| = i) \land (|N_k| = k)$ $\Rightarrow$ there exists an extension $(N_k, L_k)$ of $(N_i, L_i)$ such that $(N_k, L_k)$ k-satisfies $R$.
3. There does not exist a set of constraints $S$ satisfying properties (1) and (2) and such that $R'(N_i) \subseteq S'(N_i)$ for some $N_i$.

**Theorem 3.1.** For each consistent labeling problem $(C, N, A)$ a k-solution exists and is unique.

**Proof.** Suppose that $R$ and $S$ both satisfy properties (1) and (2). Then $R \cup S$, defined by $\{R'(N_i) \cup S'(N_i) : N_i \subseteq N, |N_i| = i, 1 \leq i \leq k\}$, also satisfies properties (1) and (2). Let $R_{\text{MAX}} = R_1 \cup \cdots \cup R_h$, where $R_1, \ldots, R_h$ are all the sets of constraints satisfying properties (1) and (2). $R_{\text{MAX}}$ is well defined because there exists at least one $R_i$ satisfying properties (1) and (2): $R_i$ such that $R_i'(N_i) = \emptyset \forall N_i$. $R_{\text{MAX}}$ also satisfies property (3), and is clearly unique. Hence a unique k-solution exists. □

Let $R_{\text{FIN}}$ be the final value of $R$ when KS terminates.

**Theorem 3.2.** $R_{\text{FIN}}$ is the k-solution to the consistent labeling problem $(C, N, A)$. 

Proof. $R_{\text{FIN}}$ satisfies property (1) because $R$ is initialized to $C$ and no labelings are added to $R$ during KS.

Furthermore, being totally relaxed, $R_{\text{FIN}}$ satisfies the following properties (a) and (b):

(a) $(L_i \in R'(N_i)) \land (i < k) \Rightarrow$ there exists an extension $(N_{i+1}, L_{i+1})$ of $(N_i, L_i)$ such that $L_{i+1} \in R'^{(i)}(N_{i+1})$ for all $N_{i+1} \supseteq N_i$ such that $|N_{i+1}| = i + 1$.

(b) $(L_i \in R'(N_i)) \land (i > 1) \Rightarrow L_{i-1} \in R'^{(i-1)}(N_{i-1})$ for all order-$(i - 1)$ sub-labelings $(N_{i-1}, L_{i-1})$ of $(N_i, L_i)$.

By applying (a) $k - i$ times, we can deduce that $R_{\text{FIN}}$ satisfies property (a'):

(a') $(L_i \in R'(N_i)) \land (N_i \subseteq N \subseteq N) \land (|N_i| = i) \land (|N_k| = k) \Rightarrow$ there exists an extension $(N_k, L_k)$ of $(N_i, L_i)$ such that $L_k \in R'(N_k)$.

Similarly, applying (b) up to $i - 1$ times shows that $R_{\text{FIN}}$ satisfies property (b'):

(b') $L_i \in R'(N_i) \Rightarrow (N_i, L_i)$ i-satisfies $R_{\text{FIN}}$.

The combination of (a') and (b') (with $i$ replaced by $k$) is precisely property (2). Therefore $R_{\text{FIN}}$ satisfies property (2).

Property (2) implies (a) and (b). (Choose $N_k = N_{i+1}$ for (a) and $N_k = N_i$ for (b).) Thus any set of constraints $S$ satisfying properties (1) and (2) must also be totally relaxed. Since relaxation is the only way labelings are eliminated from $R'(N_i)$ during KS, each $S'(N_i)$ must satisfy $S'(N_i) \subseteq R_{\text{FIN}}'(N_i)$. Thus $R_{\text{FIN}}$ satisfies property (3). $R_{\text{FIN}}$ is therefore the $k$-solution. \[\square\]

We can also use KS$(n)$ to synthesize the complete solution to the consistent labeling problem.

Corollary. When KS$(n)$ terminates, $R^n$ contains the solution to the consistent labeling problem $(C, N, A)$.

Proof. When KS$(n)$ terminates $R$ is the $n$-solution. If $R$ is an $n$-solution then, by property (2), $R^n$ satisfies each constraint $R'(N_i)$. Since $R'(N_i) \subseteq C'(N_i)$ by property (1), $R^n$ also satisfies the constraints $C$, and is maximal because of property (3). Hence, when KS$(n)$ terminates, $R^n$ is the solution to the consistent labeling problem. \[\square\]

$R$ is i-consistent [1] if, for each cardinality-$(i - 1)$ set $N_{i-1} \subset N$, for each $L_{i-1} \in R'^{i-1}(N_{i-1})$ and for each cardinality-i set $N_i \supseteq N_{i-1}$, $(N_i, L_{i-1})$ can be extended to a labeling $(N_i, L_i)$ such that $L_i \in R'(N_i)$.

Theorem 3.3. A $k$-solution is i-consistent for all $i \leq k$. 

---

AN OPTIMAL $k$-CONSISTENCY ALGORITHM 93
Proof. This follows immediately from the fact that the $k$-solution, $R_{\text{FIN}}$, satisfies property (a). □

4. Computational Complexity

The number of times the inner loop of Step 1 is executed is $\sum_{i=1}^{k} C_n^i a^i$, $C_n^i$ being the number of order-$i$ constraints and $a^i$ being the number of possible labelings in an order-$i$ constraint. An element $(N_i, L_i, i)$ is appended and hence removed from LIST at most once. Therefore the loop of Step 2A is executed no more than $\sum_{i=1}^{k-1} C_n^i a^i(n-i)a = \sum_{i=2}^{k} iC_n^i a^i$ times. The loop in Step 2B is also executed at most $\sum_{i=2}^{k} iC_n^i a^i$ times.

The arrays $R$, $M$ and $\text{Counter}$ are indexed by an ordered $i$-tuple of distinct nodes. In order to avoid wasting any memory space we use the following addressing function $f(n_1, \ldots, n_i)$ to calculate the address corresponding to the $i$-tuple $n_1 < n_2 < \cdots < n_i$:

$$f(n_1, \ldots, n_i) = \sum_{j=0}^{i-1} (C_n^{k-i} - C_n^{k-i-n_i+1})$$

where by convention $n_0 = 0$. $f(n_1, \ldots, n_i)$ is the number of ordered $i$-tuples of distinct elements, less than $(n_1, \ldots, n_i)$ in lexicographic order. The $j$th term in the sum is the number of such $i$-tuples $(a_1, \ldots, a_i)$ such that $a_1 = n_1, \ldots, a_j = n_j, n_j < a_{j+1} < n_{j+1}$. If the values of $C_k^i$ are stored in memory for all $j, k$, the calculation of $f(n_1, \ldots, n_i)$ requires $O(i)$ time. This implies that each access to $R$, $M$ or $\text{Counter}$ requires $O(i)$ operations.

The worst-case time complexity of $\text{KS}(k)$ is therefore

$$O\left(\sum_{i=1}^{k} i^2 C_n^i a^i\right) = O\left(\sum_{i=1}^{k} C_n^i a^i\right)$$

if $k$ is considered as a constant, and its space complexity is $O(\sum_{i=1}^{k} C_n^i a^i)$. In the worst case the input of the constraints $C$ requires at least $O(\sum_{i=1}^{k} C_n^i a^i)$ time and space. Thus $\text{KS}(k)$ is optimal for constant $k$ and almost optimal for all $k$.

In many consistent labeling problems only unary and binary constraints exist. As a $k$-solution algorithm, $\text{KS}$ is optimal (when $k$ is a constant and almost optimal otherwise) even if not all order-$i$ constraints exist for all $i \leq k$, since in the worst case the output of a $k$-solution requires $O(\sum_{i=1}^{k} C_n^i a^i)$ time and space. However, as a $k$-consistency algorithm, $\text{KS}$ is not necessarily optimal if order-$k$ constraints do not exist in $C$, since the output of a $k$-consistency algorithm is, in fact, $R^{k-1}$ rather than $R^k$.

We note that the use of Freuder’s network data structure [1] in $\text{KS}$, instead of arrays addressed by $f$, would give slightly different complexities: a factor $k$ greater space complexity and a factor $k$ less time complexity.
The order in which elements are removed from \( \text{LIST} \) in \( \text{KS} \) does not affect its worst-case time complexity. However, if no solution exists then it is clearly desirable to discover this fact as soon as possible. Normally the best policy is to remove elements \((N_i, L_i, i)\) from \( \text{LIST} \) in order of increasing values of \( i \). However, we could equally propagate all constraints between nodes \( \{1, \ldots, r-1\} \) before considering node \( r \), if the constraints were particularly strong for smaller values of \( r \). An adaptive choice of which constraint to propagate is also possible.

When order-\( i \) constraints exist for all \( i \leq n \), \( \text{KS}(n) \) is an almost optimal consistent labeling algorithm. It has time complexity

\[
O\left( \sum_{i=1}^{n} i^2 C_n^i a^i \right) \leq O\left( n^2 \sum_{i=1}^{n} C_n^i a^i \right) = O\left( n^2 (a+1)^n \right),
\]

whereas the input of the constraints \( C \) requires at least \( (a+1)^n \) operations. We can compare this with the time complexity of backtracking, which is at least \( O(\sum_{i=1}^{n} 2^{i-1} a^i) = O((2a)^n) \), in the worst case. In backtracking, a constraint between nodes \( n_1, \ldots, n_i \) is accessed a maximum of \( n_i \) times, but in \( \text{KS} \) the same constraint is accessed a maximum of only \( i \) times. This explains the combinatorial difference between their worst-case time complexities. We note that establishing consistency is unnecessary when \( \text{KS}(n) \) is used simply to synthesize \( R^n \); Step 2B is thus unnecessary, and even a waste of time, in \( \text{KS}(n) \).

5. Conclusion

An algorithm \( \text{KS} \) has been presented which finds the \( k \)-solution of a consistent labeling problem. It has worst-case time and space complexity which is optimal or almost optimal for all values of \( k \). It has been shown that a \( k \)-solution is \( i \)-consistent for all \( i \leq k \). \( \text{KS}(k) \), for small values of \( k \), can be used to eliminate many illegal labelings before solving the consistent labeling problem by an exhaustive search. If all order-\( i \) constraints exist for all \( i \leq n \), then \( \text{KS}(n) \) is an almost optimal algorithm for the consistent labeling problem.

REFERENCES


Received December 1988; revised version received January 1989