ELSEVIER

Information Processing Letters 82 (2002) 143–144

Information Processing Letters

www.elsevier.com/locate/ipl

Solving a system of difference constraints with variables restricted to a finite set

John P. Fishburn

Lucent Technologies Bell Laboratories, Murray Hill, NJ 07974, USA

Received 15 April 2001; received in revised form 28 June 2001
Communicated by S.E. Hambrusch

Abstract

An algorithm is given for the solution of a system of difference constraints where the variables must take on values from a given finite set of real numbers. © 2001 Published by Elsevier Science B.V.

Keywords: Algorithms; Analysis of algorithms

1. Introduction

A system of difference constraints consists of a set of variables $x_1, \ldots, x_n$ and a set of inequalities $x_j - x_i \leqslant B_{ij}$ for $1 \leqslant i, j \leqslant n$, where each B_{ij} is either a real number or $+\infty$. The Bellman–Ford algorithm [1,3,2] can be used to find a satisfying solution of a system of difference constraints, if one exists. In the original formulation, the variables are allowed to range over the real numbers. In addition, the Bellman–Ford algorithm can be used in cases where the following additional restrictions are imposed:

(1) The x_i must be chosen from a given arithmetic progression $\{\ldots, a-d, a, a+d, a+2d, \ldots\}$. In particular, the x_i may be required to be integers [2].
(2) Each x_i may have its own upper and/or lower bound imposed on it [2].
(3) The previous two conditions may be simultaneously imposed, so that all variables are restricted to a given arithmetic progression, and each x_i might also be required to satisfy its own given upper and/or lower bound.

We are interested in finding an efficient algorithm for a problem that falls into none of these categories. In our problem we are given an arbitrary finite set of allowed real values. Unlike case (3) above, however, these values are not necessarily uniformly spaced.

2. The algorithm

The proposed algorithm is a modification of the Bellman–Ford algorithm [2]. Suppose that the x_i must take on values from a given ordered finite set of real numbers $\{D_1, D_2, \ldots, D_m\}$. We first initialize each variable to the maximum value D_m. We then examine in turn each constraint $x_j \leqslant x_i + B_{ij}$, and reduce x_j by the minimum possible amount that will satisfy the constraint.

```
1 for i ← 1 to n do x_i ← D_m
2 BOOLEAN changed
3 do {
4     changed ← FALSE
```

E-mail address: jack@research.bell-labs.com (J.P. Fishburn).

0020-0190/01/$ – see front matter
PII: S0020-0190(01)00267-8

```
 5    for each constraint x_j ⩽ x_i + B_ij
 6       if(x_j > x_i + B_ij & ∃ k s.t. D_k ⩽ x_i + B_ij){
 7          x_j ← largest D_k s.t. D_k ⩽ x_i + B_ij
 8          changed ← TRUE
 9       }
10 } while (changed)
11 if any constraint is violated, return FALSE
12 return TRUE
```

3. Proof of correctness

Lemma. *Let $(S_i, \ldots, S_n)$ be any satisfying solution to the system of constraints, whose components are taken from the finite set. Then the following invariant is true throughout the execution of the algorithm*: $S_i \leqslant x_1, \ldots, S_n \leqslant x_n$.

Proof. The invariant is true at the beginning because we initialize to the largest value. We have only to show that the invariant is preserved by the "if" clause on lines 6–9. Because $(S_i, \ldots, S_n)$ is a satisfying solution, and because the invariant is satisfied before the execution of the clause, it must be the case that

$$S_j \leqslant S_i + B_{ij} \leqslant x_i + B_{ij}. \tag{1}$$

Therefore there is at least one value in the finite set, namely S_j, that is greater than or equal to S_j but less than or equal to $x_i + B_{ij}$. Since the algorithm assigns to x_j the largest of these values, the invariant must be preserved. □

Theorem (Correctness of the algorithm). *If there is a satisfying solution to the problem, the algorithm finds some satisfying solution and returns TRUE. If there is no satisfying solution, the algorithm returns FALSE.*

Proof. Suppose that there is a satisfying solution to the constraint graph, and suppose the do loop on lines 3–10 exits with an unsatisfied constraint. Then for that constraint, $x_j > x_i + B_{ij}$ and yet there is no D_k such that $D_k \leqslant x_i + B_{ij}$. But this cannot happen because $S_j \leqslant S_i + B_{ij} \leqslant x_i + B_{ij}$, so D_k could have been chosen to be S_j if nothing else.

On the other hand suppose there is no satisfying solution. Then as long as the body of the if clause is executing, there is some x_i that is being reduced during each iteration of the do loop. But each variable can only be reduced $m-1$ times, and so the algorithm will terminate and return FALSE after at most $n*(m-1)$ iterations of the do loop. □

The outer do loop cannot be executed more than $n*(m-1)$ times, and during a single traversal of the outer do loop, the inner for loop is executed once for each constraint. Thus the time complexity of the algorithm, like the Bellman–Ford algorithm, is polynomial.

An interesting generalization of the problem [4,5] would assign to each variable its own private finite set of allowed values. The proposed algorithm and proof of correctness could be slightly modified to handle this generalization.

References

[1] R.E. Bellman, On a routing problem, Quart. Appl. Math. 16 (1) (1958) 87–90.
[2] T.H. Cormen, C.E. Leiserson, R.L. Rivest, Algorithms, MIT Press, Cambridge, MA, 1990.
[3] L.R. Ford Jr., D.R. Fulkerson, Flows in Networks, Princeton Univ. Press, Princeton, NJ, 1962.
[4] S.S. Sapatnekar, Private communication, April 2001.
[5] G.J. Woeginger, Private communication, April 2001.