Complexity of solving a system of difference constraints with variables restricted to a finite set

Santiago Cifuentes  Francisco J. Soulignac  Pablo Terlisky

Universidad de Buenos Aires. Facultad de Ciencias Exactas y Naturales. Departamento de Computación, Buenos Aires, Argentina
CONICET-Universidad de Buenos Aires. Instituto de Ciencias de la Computación (ICC), Buenos Aires, Argentina

Abstract

Fishburn developed an algorithm to solve a system of $m$ difference constraints whose $n$ unknowns must take values from a set with $k$ real numbers [Solving a system of difference constraints with variables restricted to a finite set, Inform Process Lett 82 (3) (2002) 143–144]. We provide an implementation of Fishburn’s algorithm that runs in $O(n + km)$ time.

1 Introduction

A system of difference constraints is a pair $(X, S)$ where $X$ is a set of unknowns $x_1, \ldots, x_n$ and $S$ is a family of $m$ constraints of the form $e_{ij}: x_i - x_j \leq b_{ij}$, for $b_{ij} \in \mathbb{R}$. Throughout this article, we refer to $s: X \rightarrow \mathbb{R}$ as being a solution to $(X, S)$. Moreover, we say that $s$ is restricted to $D \subseteq \mathbb{R}$ when $s(x) \in D$ for every $x \in X$. If $s(x_i) - s(x_j) \leq b_{ij}$ for $e_{ij} \in S$, then $s$ satisfies $e_{ij}$, while if $s$ satisfies every constraint in $S$, then $s$ is feasible. The system $(X, S)$ itself is feasible (resp. restricted to $D \subseteq \mathbb{R}$) when it has a feasible solution (resp. restricted to $D$).

It is well known that the feasibility of $(X, S)$ can be decided in $O(nm)$ time with the Bellman-Ford algorithm [1]. The output of Bellman-Ford is either a feasible solution or a subset of constraints that admits no solution. In [3], Fishburn proposed a simple algorithm to decide if $(X, S)$ is feasible restricted to a set $D = \{d_1 < \ldots < d_k\}$, that outputs a feasible solution $s$ restricted to $D$ in the affirmative case. Fishburn’s algorithm can be rephrased as in Algorithm 1.

Fishburn observed that loop 3–6 is executed $O(kn)$ times, because $s(x_i)$ is decreased at Step 6. Consequently, as a single traversal of $S$ is enough to find an unsatisfied constraint $e_{ij}$ at Step 3, Algorithm 1 runs in $O(kmn)$ time.

Fishburn did not provide a faster implementation of his algorithm in [3]. Yet, it is easy to see that the update of $s(x_i)$ at Step 6 only affects the satisfiability of those constraints $e_{\ell i} \in S$, $1 \leq \ell \leq n$. Hence, Step 3 can be restricted to a small subset of $S$, improving the efficiency of the algorithm. In this note we take advantage of this fact to show an implementation of Fishburn’s algorithm that runs in $O(n + km)$ time.

Since its appearance in late 2001, Fishburn’s algorithm was applied mainly for the optimization of clock skew in digital circuits [2]. In this domain, the canonical application of Fishburn’s algorithm is the clock shift decision problem, introduced in early 2002 by Singh and Brown [7]:

* * *
for a fixed clock period \( t \) and a finite set \( D \) of clock shifts (i.e., delays), determine there exists an assignment of clock shifts to the registers of a digital circuit that satisfies the double-clocking and zero-clocking constraints [2]. To solve this problem, Singh and Brown apply a “discrete version of the Bellman-Ford algorithm” [7, Algorithm CSDPcore], that is nothing else than a restatement of Fishburn’s algorithm. Even though Singh and Brown explicitly state that Fishburn’s algorithm runs in \( O(kmn) \) time, they also state that they “implement the algorithm in such a way that searching for an unsatisfied constraint takes at most \( O(|R|) \) time”, where \( R \) is the set of unknowns [7, p. 124]. Interestingly, later works that apply Fishburn’s algorithm to the clock shift decision problem mention that its time complexity is \( O(kmn) \) [e.g. 4–6]. In particular, an \( O(n + m) \) time algorithm that works only for circuits with \( k = 2 \) registers was developed in [4], which is based on a reduction of the problem to 2-SAT. Our implementation of Fishburn’s algorithm generalizes this result, without requiring an implementation of 2-SAT, as it runs in \( O(n + m) \) time for every constant \( k \).

2 An efficient implementation

Algorithm 2 is the improved version of Algorithm 1 that we propose. The main difference between both implementations is that the Algorithm 2 keeps a set of unknowns \( U \). Throughout the lifetime of the algorithm, an unknown \( x_i \) belongs to \( U \) if and only if some constraint \( e_{ij} \in S \) is not satisfied by the current solution \( s \). Hence, \( U \) gives immediate access to those constraints that are not satisfied by \( s \).

**Theorem 1.** Let \((X, S)\) be a system with \( m \) difference constraints and \( n \) unknowns, and \( D \) be a set with \( k \) real numbers \( d_1 < \ldots < d_k \). The following statements are true when Algorithm 2 is executed with input \((X, S)\) and \( D \subseteq \mathbb{R} \):

a) Algorithm 2 stops in \( O(n + km) \) time.

b) If Algorithm 2 returns a solution \( s \neq \perp \), then \( s \) is a feasible solution to \((X, S)\) restricted to \( D \).

c) If \((X, S)\) is feasible restricted to \( D \), then Algorithm 2 returns a solution \( s \neq \perp \).

**Proof.** Before dealing with a–c, we prove that the following statements are true immediately before each execution of Step 7:

(i) \( s \) is a solution to \((X, S)\) restricted to \( D \),

(ii) if \( s' \) is a feasible solution to \((X, S)\) restricted to \( D \), then \( s'(x) \leq s(x) \) for every \( x \in X \), and

(iii) \( x_j \in U \) (\( 1 \leq j \leq n \)) if and only if \( s \) does not satisfy some constraint in \( S_j^+ \).
Algorithm 2 Improved Fishburn’s Algorithm

Input: A system of difference constraints \((X, S)\) and a set of real numbers \(D = \{d_1 < \ldots < d_k\}\)  
Output: a feasible solution \(s \) to \((X, S)\) restricted to \(D\) or \(\bot\) if such a solution does not exist  

1. let \(s\) be a function in \(X \to \mathbb{R}\)  
2. for \(x \in X: s(x) := d_k\)  
3. for \(i := 1, \ldots, n;\)  
4. let \(S_i^+ := \{e_{ij} \in S \mid 1 \leq j \leq n\}\)  
5. let \(S_i^- := \{e_{ji} \in S \mid 1 \leq j \leq n\}\).  
6. let \(U := \{x_i \in X \mid (\exists e_{ij} \in S_i^+)(s \text{ does satisfy } e)\}\).  
7. while \(U \neq \emptyset;\)  
8. Remove some unknown \(x_i\) from \(U\).  
9. if \((\exists e_{ij} \in S_i^+)(d_1 - s(x_j) > b_{ij});\)  
10. return \(\bot\)  
11. \(s(x_i) := \max\{d \in D \mid (\forall e_{ij} \in S_i^+)(d - s(x_j) \leq b_{ij})\}\).  
12. for all \(e_{ji} \in S_i^-\) not satisfied by \(s:\)  
13. Insert \(x_j\) into \(U\)  
14. return \(s\)

Certainly, (i)–(iii) hold immediately before the first execution of Step 7, because of Steps 2–6. Statement (i) remains true because \(s\) is updated only by Step 11. Regarding (ii), let \(s'\) be a feasible solution to \((X, S)\) restricted to \(D\). By hypothesis, (ii) is true before Step 7, thus \(s'(x_i) \leq s(x_i)\). Then, taking into account that \(S_i^+\) is precisely the subset of constraints in \(S\) that have \(x_i\) with coefficient 1 (Step 4), it follows that \(s'(x_i) \leq b_{ij} + s'(x_j) \leq b_{ij} + s(x_j)\) for every \(e_{ij} \in S_i^+\). Consequently, \(s'(x_i) \leq s(x_i)\) after Step 11 and, therefore, (ii) is true immediately before the next execution of Step 7. Finally, regarding (iii), observe that \(s\) satisfies every constraint in \(S_i^+\) after Step 11, thus (iii) holds for \(j = i\) by Step 8. Moreover, \(e_{ji} \in S_i^+\) (1 \(\leq \ell \leq n\)) is not satisfied by \(s\) after Step 11 if and only if either \(\ell \neq i\) and \(e_{ij}\) was neither satisfied by \(s\) before Step 11 or \(\ell = i\) and \(e_{ji}\) is not satisfied by \(s\) after its update at Step 11. By (iii), the former happens if and only if \(x_j \in U\) before Step 13, while the latter happens if and only if \(e_{ji} \in S_i^-\) (Step 5) is not satisfied by \(s\), in which case \(x_j\) is inserted into \(U\) at Step 13. Altogether, (iii) is also true before each execution of Step 7.

In what follows, we use (i)–(iii) to prove a–c.

a) Suppose \(x_i\) (1 \(\leq i \leq n\)) is removed from \(U\) at some execution of Step 8. By (iii), some constraint \(e_{ij} \in S_i^+\) is not satisfied by \(s\), thus \(s(x_i) - s(x_j) > b_{ij}\). Consequently, either \(d_1 - s(x_j) > b_{ij}\) and Algorithm 2 halts at Step 10 or \(s(x_i)\) is updated to \(d\) at Step 11 for some \(d_1 \leq d < s(x_i)\). Whichever the case, \(x_i\) is removed at most \(O(k)\) times from \(U\), thus Algorithm 2 runs for a finite amount of time.

Regarding the time consumed by Algorithm 2, observe that each iteration of the Loop 7–13 requires \(O(|S_i^+| + |S_i^-| + p - q)\) time with a standard implementation, where

- \(d_p\) is the value of \(s(x_i)\) before the execution of 11,
- \(q = 0\) if the loop reaches Step 10 and the algorithm halts, and
- \(q > 0\) and \(d_q\) is the value assigned to \(s(x_i)\) if Step 11 is reached by the loop.

Then, as Steps 2–6 require \(O(n + m)\) time and each \(x \in X\) is removed at most \(O(k)\) times from \(U\), the Handshaking Lemma implies that the total time consumed by Algorithm 2 is:

\[
O \left( n + m + \sum_{i=1}^{n} k \left( |S_i^+| + |S_i^-| \right) \right) = O(n + km).
\]
b) Algorithm 2 returns a solution \( s \neq \bot \) only if Step 14 is executed, a situation that only occurs if \( U = \emptyset \) when Step 7 is executed for the last time. By (i), \( s \) is a solution to \( (X, S) \) restricted to \( D \), and, by (iii), \( s \) is feasible because \( S = S^+_1 \cup \ldots S^+_n \).

c) If Algorithm 2 returns \( \bot \), then Step 10 is reached. By Step 9, \( d_1 - s(x_j) > b_{ij} \) for some \( e_{ij} \in S^+_i \), where \( x_i \) is the unknown removed from \( U \) at Step 8. If \( s' \) is a solution to \( (X, S) \) restricted to \( D \), then either \( s'(x_j) > s(x_j) \) and \( s' \) is not feasible by (ii) or \( d_1 \leq s'(x_j) \leq s(x_j) \) and \( s' \) does not satisfy \( e_{ij} \) because \( d_1 - s'(x_j) \geq d_1 - s(x_j) > b_{ij} \). Whichever the case, \( s' \) is not feasible.

References

[1] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to algorithms*. MIT Press, Cambridge, MA, third edition, 2009.

[2] J. Fishburn. Clock skew optimization. *IEEE Trans Comput*, 39(7):945–951, 1990. doi:10.1109/12.55696.

[3] J. P. Fishburn. Solving a system of difference constraints with variables restricted to a finite set. *Inform Process Lett*, 82(3):143–144, 2002. doi:10.1016/S0020-0190(01)00267-8.

[4] Y. Kohira and A. Takahashi. 2-SAT based linear time optimum two-domain clock skew scheduling in general-synchronous framework. *IEICE Trans Fundam Electron Commun Comput Sci*, 97(12):2459–2466, 2014.

[5] L. Li, Y. Lu, and H. Zhou. Optimal and efficient algorithms for multidomain clock skew scheduling. *IEEE Trans Very Large Scale Integr VLSI Syst*, 22(9):1888–1897, 2014. doi:10.1109/TVLSI.2013.2280732.

[6] H. Mashiko. *Digital LSI Design Methods Considering Process Variations in Advanced Technology Nodes*. PhD thesis, The University of Aizu, 2017.

[7] D. P. Singh and S. D. Brown. Constrained clock shifting for field programmable gate arrays. In *Proceedings of the 2002 ACM/SIGDA Tenth International Symposium on Field-Programmable Gate Arrays*, FPGA ’02, pages 121–126, New York, NY, USA, 2002. ACM. doi:10.1145/503048.503067.