Electronic Colloquium on Computational Complexity, Report No. 127 (2015)

On the Optimality of Bellman–Ford–Moore Shortest Path Algorithm[☆]

S. Jukna[a,1], G. Schnitger[a]

[a]Institute of Computer Science, Goethe University, Frankfurt am Main, Germany

Abstract

We prove a general lower bound on the size of branching programs over any semiring of zero characteristic, including the $(\min, +)$ semiring. Using it, we show that the classical dynamic programming algorithm of Bellman, Ford and Moore for the shortest *s*-*t* path problem is optimal, if only Min and Sum operations are allowed.

Key words: Computational complexity, shortest paths, matrix multiplication, dynamic programming, tropical semiring, lower bounds

1. Introduction

Dynamic programming algorithms for discrete minimization problems are actually (recursively constructed) circuits or branching programs over the $(\min, +)$ semiring, known also as *tropical* semiring. So, in order to understand the limitations of the dynamic programming, we need lower-bounds arguments for tropical circuits and branching programs.

In this note, we present such an argument for tropical branching programs. These programs correspond to dynamic programming algorithms solving minimization problems $f : \mathbb{N}^n \to \mathbb{N}$ of the form

$$f(x_1, \ldots, x_n) = \min_{a \in A} \sum_{i=1}^{n} a_i x_i \,, \tag{1}$$

where $A \subset \mathbb{N}^n$ is a finite set of nonnegative integer vectors $a = (a_1, \ldots, a_n)$. We prove that every tropical branching program solving f must have at least $f(1, \ldots, 1) \cdot w(f)$ edges, where $w(f)$ is the smallest size of a subset $S \subseteq [n] = \{1, \ldots, n\}$ such that, for every vector $a \in A$, there is a position $i \in S$ with $a_i \neq 0$ (Lemma 3). We then demonstrate this general lower bound by two almost optimal lower bounds.

Shortest paths. Our first application concerns the classical dynamic programming algorithm of Ford [5], Moore [10], and Bellman [1] for the shortest *s*-*t* path problem. This algorithm actually solves the *shortest k-walk* problem: given an assignment of nonnegative weights to the edges of the complete graph on $[n] = \{1, \ldots, n\}$, find the minimum weight of a walk of length k from node $s = 1$ to the node $t = n$. Recall that a *walk* of length k is an alternating sequence of $k+1$ nodes and connecting edges. A walk can travel over any node (except of s and t) and any edge (including loops) any number of times. A *path* is a walk which cannot travel over any node more than once.

In a related *shortest k-path* problem, the goal is to compute the minimum weight of an *s*-*t* path of length *at most* k. Note that, if we give zero weight to all loops, then these two problems are equivalent. This holds because weights are nonnegative, every *s*-*t* walk of length k contains an *s*-*t* path of length $\leqslant k$, and every *s*-*t* path of length $\leqslant k$ can be extended to an *s*-*t* walk of length k by adding loops.

The Bellman–Ford–Moore algorithm gives a tropical branching program with kn nodes and kn^2 edges solving the k-walk problem (see Lemma 4 below). By combining our general lower bound with

[☆]Research supported by the DFG grant SCHN 503/6-1.

[1]Affiliated with Institute of Mathematics and Informatics, Vilnius University, Vilnius, Lithuania.

ISSN 1433-8092

a result of Erdős and Gallai [3] about the maximal number of edges in graphs without long paths, we show (Theorem 1) that this algorithm is almost optimal, if only Min and Sum operations are allowed: at least about $kn(n-k)$ edges are also necessary in *any* tropical branching program solving the k-walk problem.

Matrix multiplication. Our next application concerns the complexity of matrix multiplication over the $(\min,+)$ semiring. Kerr [7] has shown that any $(\min,+)$ circuit, simultaneously computing all the n^2 entries of the product of two $n\times n$ matrices over the $(\min,+)$ semiring, requires $\Omega(n^3)$ gates. This showed that the dynamic programming algorithm of Floyd [4] and Warshall [16] for the all-pairs shortest paths problem is optimal, if only Min and Sum operations are allowed. Later, Pratt [14], Paterson [12], and Mehlhorn and Galil [9] independently proved the same lower bound even over the boolean semiring. This showed that the dynamic programming algorithm of Floyd [4] and Warshall [16] for the all-pairs shortest paths problem is optimal, if only Min and Sum operations are allowed.

These lower bounds, however, do not imply the same lower bound for the *single-output* version M_n of this problem: compute the sum of all entries of the product matrix. Using our general lower bound, we show that the minimum number of contacts in a branching program solving M_n over the $(\min,+)$ semiring is $2n^3$ (Theorem 2).

Remark 1. Let us stress that we are interested in proving *small* lower bounds: in both problems above, we have $N=\Theta(n^2)$ variables, and the bounds have, respectively, the form $\Omega(kN)$ and $\Omega(N^{3/2})$. On the other hand, *large*, even exponential in N, lower bounds for some other problems are much easier to obtain. For example, it is relatively easy to show (see, e.g. [6, Theorem 30]) that tropical branching programs of exponential size are required to solve the minimum weight spanning tree, or the minimum weight perfect matching problems. An argument allowing to prove up to N^2 lower bounds for monotone boolean branching programs was suggested by Moore and Shannon [11], and Markov [8] (Lemma 2). Our general lower bound $f(1,\dots,1)\cdot w(f)$ on the size of tropical branching programs solving a minimization problem f is an adoption of their argument.

2. Branching programs and their polynomials

Let $(R,+,\times,0,1)$ be a semiring with a "sum" $(+)$ and "product" $(\times)$ operations. Recall that a (multivariate) polynomial over R is a formal expression of the form

$$f(x_1,\dots,x_n)=\sum_{a\in A}c_a\prod_{i=1}^{n}x_i^{a_i}\,, \tag{2}$$

where $A\subset\mathbb{N}^n$ is a finite set of nonnegative integer vectors, and $c_a\geqslant 1$ are integer coefficients. The *degree* of a monomial $p=\prod_{i=1}^n x_i^{a_i}$ is the sum $a_1+a_2+\cdots+a_n$ of its exponents. The *support* of p is the set $X_p=\{x_i\colon a_i\neq 0\}$ of all variables occurring in the monomial with nonzero degree. The *degree* of a polynomial is the minimum degree of its monomial. A monomial of f is *minimal*, if its support does not contain the support of any another monomial of f as a proper subset. Let $\mathrm{Sup}(f)$ denote the family of supports of all minimal monomials of f.

Every polynomial f *defines* the function $f\colon R^n\to R$, whose value $f(r)=f(r_1,\dots,r_n)$ is obtained by substituting elements $r_i\in R$ for x_i in f. Over different semirings R, these functions may be different. For example, in the *boolean* semiring, we have $R=\{0,1\}$, $x+y:=x\vee y$, $x\times y:=x\wedge y$, $0:=0$, and $1:=1$, whereas in the *tropical* semiring, we have $R=\mathbb{N}\cup\{+\infty\}$, $x+y:=\min\{x,y\}$, $x\times y:=x+y$, $0:=\infty$, and $1:=0$. Hence, over these two semirings, the functions defined by the polynomial (2) are, respectively,

$$f=\bigvee_{a\in A}\ \bigwedge_{i\colon a_i\neq 0}x_i \quad\text{and}\quad f=\min_{a\in A}\sum_{i\colon a_i\neq 0}a_ix_i\,.$$

A semiring $(R,+,\times,0,1)$ is of *zero characteristic*, if $1+1+\cdots+1\neq 0$ holds for any finite sum of the unity 1. Note that both semirings above are such.

Lemma 1. *If two polynomials f and g define the same function over a semiring of zero-characteristic, then* $\mathrm{Sup}(f) = \mathrm{Sup}(g)$.

Proof. Let us first show the following auxiliary claim.

Claim 1. The support of every monomial of g must contain the support of at least one monomial of f, and vice versa.

Proof. Assume contrariwise that there is a monomial q of g such that $X_p \setminus X_q \neq \emptyset$ holds for all monomials p of f. If we set to 1 all variables in X_q, and set to 0 all the remaining variables, then on the resulting assignment a, we have that $f(a) = 0$, because every monomial of f contains at least one variable set to 0. But the monomial q of g is evaluated to 1. Since the semiring is of zero-characteristic, this yields $g(a) \neq 0$, a contradiction. □

Assume now that $\mathrm{Sup}(f) \neq \mathrm{Sup}(g)$. Then, by symmetry, we may assume that there is a minimal monomial p of f such that $X_q \neq X_p$ holds for all monomials q of g. Then, for every q, we have either $X_q \setminus X_p \neq \emptyset$, or $X_q \subset X_p$ (proper inclusion). By Claim 1, the latter inclusion is impossible, since the monomial p is minimal in f. Thus, we have that $X_q \setminus X_p \neq \emptyset$ must hold for all monomials q of g. If we set to 1 all variables in X_p, and set to 0 all the remaining variables, then on the resulting assignment a, we have that $g(a) = 0$. But since $q(a) = 1$ and the semiring is of zero-characteristic, we have that $f(a) \neq 0$, a contradiction. □

By a *branching program* with variables $x_1, \dots, x_n$ we will mean a directed acyclic graph G with two specified nodes, the source node a and the target node b. Paths from a to b are called *chains*. Each edge is either unlabeled (is a *rectifier*) or is labeled by some variable (is a *contact*). The graph may be a multigraph, that is, several edges may have the same endpoints. The *size* of a program is the total number of contacts, and the *depth* is the maximum number of edges in a chain.

Every branching program G *produces* a unique polynomial f_G in a natural way. Namely, each chain π in G defines a monomial M_π, which is just the product of labels of contacts along π. The polynomial f_G is then the sum of monomials M_π over all chains in G:

$$f_G(x_1, \dots, x_n) = \sum_{\pi \text{ is a chain in } G} M_\pi \,.$$

The branching program G *computes* a given polynomial f over a semiring R, if both polynomials f_G and f define the same function over R. Every polynomial f can be computed by a trivial branching program which has a separate chain for each monomial of f. However, some polynomials allow much more compact representation as branching programs (see Fig. 1).

Remark 2. It is well known (see, e.g. [6, Lemma 11]) that, for every polynomial f, every branching program computing f over a semiring of zero characteristic must also compute f over the boolean semiring $(\{0,1\}, \vee, \wedge, 0, 1)$. Thus, every lower bound on the size of monotone boolean branching programs, i.e. programs over the boolean semiring, also holds for branching programs over any semiring of zero characteristic.

3. A general lower bound

Our starting point is the following lower bound on the size of branching programs in terms of their "length" and "width".

Lemma 2 (Moore–Shannon [11], Markov [8])**.** *If every chain in a branching program has at least l contacts, and if at least w contacts must be removed to destroy all chains, then there must be at least $l \cdot w$ contacts.*

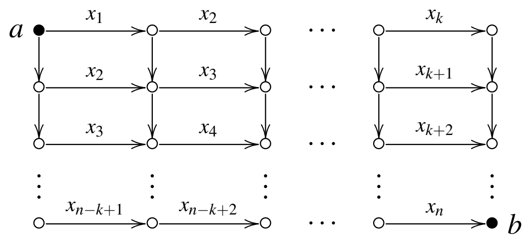

Figure 1: A branching program computing the elementary symmetric polynomial

$$f(x_1,\dots,x_n)=\sum_{|S|=k}\prod_{i\in S}x_i$$

over any semiring. The polynomial has $\binom{n}{k}$ monomials, but the program has only $k(n-k+1)$ contacts. Over the $(\min,+)$ semiring, this polynomial corresponds to the minimization problem

$$f(x_1,\dots,x_n)=\min_{|S|=k}\sum_{i\in S}x_i\,.$$

In particular, if a monotone boolean function f has no minterm shorter than l and no maxterm shorter than w, then every branching program computing f over the boolean $(\vee,\wedge)$ semiring must have at least $l\cdot w$ contacts. This, for example, implies that optimal branching programs over this semiring for the threshold-k function of n variables (a boolean version of the elementary symmetric polynomial given in Fig. 1) have exactly $k(n-k+1)$ contacts.

Unfortunately, the boolean version of the k-walk problem has very short minterms, and Lemma 2 cannot yield any nontrivial lower bound over the boolean semiring. Even worse, we show in Sect. 6 that there is no analogue of this lemma in the presence of short chains: then a branching program may have much fewer than $l\cdot w$ contacts.

Still, we will now prove a version of this lemma allowing us to show that Bellman–Ford–Moore is optimal at least over the tropical $(\min,+)$ semiring.

Let $f(x_1,\dots,x_n)$ be a polynomial. To make our argument flexible in applications, we assume that some of the variables $x_1,\dots,x_n$ are declared as *good*, and the remaining as *bad*.

After this declaration, define the *degree* of a monomial as the sum of exponents of its good variables. Then the degree $\deg(f)$ of a polynomial f is the minimum degree of its monomial. If we treat polynomials over some fixed semiring R, then it is natural to define the *semantical degree* $d(f)$ of f to be the smallest degree of a polynomial defining the same function as f over R.

Remark 3. Over the boolean $(\vee,\wedge)$ semiring, f is a monotone boolean function, and $d(f)$ is the minimum number of good variables such that setting to 1 these variables and all bad variables, forces f output 1, independently on the values of other variables.

Over the $(\min,+)$ semiring, the degree $\deg(f)$ of f is the value $f(\alpha)$ of f on the assignment α which gives weight 1 to all good variables, and zero weight to the remaining variables. Hence, over this semiring, we have that $d(f)=\deg(f)=f(\alpha)$.

Define the *width* $w(f)$ of a polynomial f to be the minimum number of good variables such that every monomial of f contains at least one of these variables. In other words, $w(f)$ is the minimum number of variables such that setting these variables to 0 forces f to output 0 independently on the values of the remaining variables.

Lemma 3 (Main Lemma)**.** *Every branching program G computing a polynomial f over a semiring of zero characteristic must have at least $d(f)\cdot w(f)$ contacts.*

Proof. If $d(f)=0$, there is nothing to prove. So, we can assume that $d(f)\geqslant 1$, and hence, also $w(f)\geqslant 1$. Say that an edge e of G is *good*, if it is a contact labeled by a good variable; hence, rectifiers as well as contacts labeled by bad variables are *bad*.

Claim 2. Every chain in G has at least $d(f)$ good contacts.

Proof. Let l be the smallest number of good contacts in a chain of G. Since the program G computes f, the polynomial f_G produced by G must define the same function as f. Hence, $d(f)$ is at most the degree l of f_G, as claimed. □

A *cut* in a branching program is a set C of its good contacts such that every chain in G contains at least one contact in C.

Claim 3. There are at least $d(f)$ disjoint cuts in G.

Proof. Associate with every node u in G the minimum number $l(u)$ of good contacts in a path from the source node a to u. By Claim 2, the target node b has $l(b)\geqslant d(f)$. Moreover, $l(v)\leqslant l(u)+1$ holds for every edge $e=(u,v)$, and $l(v)\leqslant l(u)$ if the edge e is bad. Let C_i be the set of all edges (u,v) such that $l(u)=i$ and $l(v)=i+1$. Since the C_i are clearly disjoint, and all edges in C_i must be good, it remains to show that each C_i is a cut.

To show this, take an arbitrary chain $(u_1,u_2,\ldots,u_m)$ with $u_1=a$ and $u_m=b$. The sequence of numbers $l(u_1),\ldots,l(u_m)$ must reach the value $l(b)\geqslant d(f)$ by starting at $l(a)=0$. At each step, the value can be increased by at most $+1$. So, there must be a j where a jump from $l(u_j)=i$ to $l(u_{j+1})=i+1$ happens, meaning that the edge (u_j,u_{j+1}) belongs to C_i, as desired. □

Claim 4. Every cut in G has at least $w(f)$ contacts.

Proof. Let C be a cut in G, and let X_C denote the set of good variables labeling the contacts in C. It is enough to show that every monomial p of f must contain at least one variable in X_C, because then $|C|\geqslant|X_C|\geqslant w(f)$.

The support X_p of p must contain the support X_q of some minimal monomial q of f. By Lemma 1, there must be a chain π in G, the set of whose labels coincides with X_q. Since C is a cut, ar least one good contact of π must belong to C; the label $x_i\in X_C$ of this contact belongs then to X_q, and hence, also to X_p. □

Now, by Claim 3, the branching program G must have at least $d(f)$ disjoint cuts. By Claim 4, each of these cuts must have at least $w(f)$ contacts. Hence, the program must have at least $d(f)\cdot w(f)$ contacts, as claimed. □

4. Bellman–Ford–Moore

The *k-walk polynomial* $W_{n,k}$ has one variable $x_{i,j}$ for each edge $\{i,j\}$ of the complete graph K_n. Each its monomial corresponds to a walk of length k, and has the form $x_{1,i_1}x_{i_1,i_2}\cdots x_{i_{k-2},i_{k-1}}x_{i_{k-1},n}$ for not necessarily distinct nodes $i_1,\ldots,i_{k-1}$ in $\{2,\ldots,n-1\}$. That is, we assume that each node, except of 1 and n, has a loop.

Lemma 4 (Bellman [1], Ford [5], Moore [10])**.** *Over any semiring, the polynomial $W_{n,k}$ can be computed by a branching program of depth k with at most kn nodes and at most kn^2 edges.*

Proof. The dynamic programming algorithm of Bellman, Ford and Moore is amazingly simple. It computes $W_{n,k}$ by recursively computing the polynomials $F_j^{(l)}$ whose monomials correspond to walks of length l from node 1 to node j. It first sets $F_j^{(1)}=x_{1,j}$ for all $j=2,\ldots,n-1$, and uses the recursion

$$F_j^{(l+1)}=\sum_{i=2}^{n-1}F_i^{(l)}\times x_{i,j}\,.$$

To construct the desired branching program, arrange the nodes of a branching program into $k+1$ layers of nodes $V_0, V_1, \ldots, V_k$, where $V_0 = \{a\}$, $V_k = \{b\}$ and $|V_1| = \ldots = |V_{k-1}| = n-2$; each V_i for $i = 1, \ldots, k-1$ is a disjoint copy of the set of nodes $\{2, \ldots, n-1\}$. Edges go only from one layer to the next layer. The j-th node on the $(l+1)$-th layer is entered by a contact labeled by $x_{i,j}$ from the i-th node on the previous l-th layer. The program has $(k-1)(n-2)+2 \leqslant kn$ nodes and $(k-2)(n-2)^2 + 2(n-2) \leqslant kn^2$ edges. $\square$

Remark 4. The *s-t* connectivity function STCON(n) is a monotone boolean function which, given a (directed or undirected) graph on n nodes, accepts this graph if and only if it has a path from s to t. Lemma 4 implies that both directed and undirected versions of this problem can be solved by a monotone boolean branching program with at most n^3 edges. Another classical model for computing boolean functions is that of *switching networks*. The only difference of this model from branching programs is that now the underlying graph is *undirected*. Interestingly, in the monotone setting, this later model can be super-polynomially weaker than that of branching programs. Namely, Potechin [13] has shown that every monotone switching network for the directed version of STCON(n) must have $n^{\Omega(\log n)}$ edges.

We now show that, over the $(\min,+)$ semiring, the upper bound given by Lemma 4 cannot be substantially improved. Over this semiring, $W_{n,k}$ turns into a minimization problem

$$W_{n,k} = \min\left\{x_{1,i_1} + x_{i_1,i_2} + \cdots + x_{i_{k-1},n}\right\}. \tag{3}$$

To spare parenthesis, we say that a function $f(n)$ is *at least about* $g(n)$, if $f(n) = \Omega(g(n))$.

Theorem 1. *Every branching program computing $W_{n,k}$ over the $(\min,+)$ semiring requires at least about $kn(n-k)$ contacts.*

Proof. Call a variable $x_{i,j}$ of $f = W_{n,k}$ *good*, if $i, j \notin \{s,t\}$; recall that $s = 1$ is the start node, and $t = n$ the target node in K_n. Thus, good variables $x_{i,j}$ correspond to the edges of the complete graph K_{n-2} on $\{2, \ldots, n-1\}$. Recall that, over the $(\min,+)$ semiring, the semantical degree $d(f)$ of a polynomial f is just its value $f(\alpha)$ on the assignment α which sets all good variables to 1, and the rest to 0 (see Remark 3). Since every sum in (3) has only two bad variables, we have that $d(f) \geqslant k-2$. To lower bound the width $w(f)$, we will use the following result of Erdős and Gallai [3, Theorem 2.6]:

- At least $m(m-l)/2$ edges must be removed from K_m in order to destroy all paths of length $l \geqslant 1$.

Now let Y be a set of $|Y| = w(f)$ good variables of f such that every sum of f contains at least one of them. For every path $p = (i_1, \ldots, i_{k-1})$ of length $k-2$ in K_{n-2}, there is a sum $x_{s,i_1} + x_{i_1,i_2} + \cdots + x_{i_{k-2},i_{k-1}} + x_{i_{k-1},t}$ in f. This sum must contain at least one variable $x_{u,v}$ in Y. Since this variable must be good, we have that $x_{u,v}$ must be distinct from x_{s,i_1} and $x_{i_{k-1},t}$, that is, $\{u,v\}$ must be an edge of the path p. Thus, removal from K_{n-2} of all edges corresponding to variables in Y destroys all paths of length $k-2$ in K_{n-2}. When applied with $m = n-2$ and $l = k-2$, the Erdős–Gallai theorem implies that $w(f) = |Y| \geqslant (n-2)(n-k)/2$. Since $d(f) \geqslant k-2$, Lemma 3 implies that every tropical branching program computing f must have at least $d(f) \cdot w(f)$ contacts, which is at least about $kn(n-k)$. $\square$

5. Matrix multiplication

We now consider the problem of computing the sum of all entries of the product of two matrices over the tropical semiring:

$$M_n(x,y) = \sum_{i,j \in [n]} \min_{k \in [n]} \{x_{i,k} + y_{k,j}\}.$$

Theorem 2. *The minimum number of contacts in a branching program computing M_n over the $(\min,+)$ semiring is $2n^3$.*

Proof. The upper bound $2n^3$ is trivial, since each minimum $g_{i,j} = \{x_{i,k} + y_{k,j}\}$ can be computed using a bunch of $2n$ contacts. To prove the lower bound, we will again use Lemma 3. This time we declare all variables of $f = M_n$ as *good*. Since $f(1,1,\dots,1) = 2n^2$, the semantical degree of f is $d(f) \geqslant 2n^2$. On the other hand, in order to force f to output ∞, there must be at least one pair $i, j \in [n]$ such that the minimum $g_{i,j}$ outputs ∞. Thus, at least n variables must be set to ∞, implying that $w(f) \geqslant n$. By Lemma 3, any tropical branching program computing $f = M_n$ must have at least $d(f) \cdot w(f) \geqslant 2n^3$ contacts. □

6. Concluding remarks

We presented a general lower bound for $(\min, +)$ branching programs solving minimization problems with linear target functions. We then used it to show that the Bellman–Ford–Moore dynamic programming algorithm for the shortest *s*-*t* path problem, as well as a trivial matrix multiplication algorithm over the $(\min, +)$ semiring are essentially optimal, if only Min and Sum operations are allowed.

The most interesting in this context problem is to extend these lower bounds to branching programs over the *boolean* $(\vee, \wedge)$ semiring, i.e. to boolean monotone branching programs. The reason why Lemma 2 cannot yield any nontrivial lower bound for the boolean versions of the considered polynomials is that then their semantical degree is small: $d(f) \leqslant 3$ for $f = W_{n,k}$, and $d(f) \leqslant 2n$ for $f = M_n$.

When stated differently, Lemma 2 says that, if all chains in a program have length at least l, then it is enough to remove an at most a $1/l$ fraction of all edges to destroy all chains. But what happens, if short chains may be present – is it then also enough to remove an at most about a fraction $1/l$ of the edges to destroy all chains of length l or longer? Shorter chains, as well as long paths between other nodes may then survive.

Unfortunately, there exists no analogue of Moore–Shannon–Markov lemma (Lemma 2) for branching programs with short chains: then even a *constant* fraction of edges may be necessary to remove, even for large path-lengths l. Namely, a sequence of directed acyclic graphs H_n of constant maximum degree d on $n = m2^m$ nodes is constructed in [15] with the following property:

- For every constant $0 \leqslant \varepsilon < 1$ there is a constant $c > 0$ such that, if any subset of at most cn nodes is removed from H_n, the remaining graph contains a path of length at least $2^{\varepsilon m}$.

Take now two new nodes a and b, and draw an edge from a to every node of H_n, and an edge from every node of H_n to b. The resulting graph G_n still has at most $2n + dn = O(n)$ edges, and has the property:

- For every constant $0 \leqslant \varepsilon < 1$, there is a constant $c' > 0$ such that, if any subset of at most $c'n$ edges is removed from G_n, the remaining graph contains an *a*-*b* path with $2^{\varepsilon m}$ or more edges.

Proof. Call the nodes of H_n *inner* nodes of G_n. Remove any subset of at most $c'n$ edges from G_n, where $c' = c/2$. After that, remove an inner node if it was incident to a removed edge. Note that at most $2c'n = cn$ inner nodes are then removed. None of the edges incident to survived nodes was removed. In particular, each survived inner node is still connected to *both* nodes a and b. By the above property of H_n, there must remain a path of length $2^{\varepsilon m}$ consisting entirely of survived inner nodes. Since the endpoints of each of these paths survived, each of them can be extended to an *a*-*b* path in G_n. □

In branching programs considered above, contacts are labeled by single variables. One can extend the model of $(\min, +)$ branching programs by allowing the labels of contacts to be arbitrary linear combinations $\sum_{i \in S} a_i x_i$ with integer coefficients. Albeit the Bellman–Ford–Moore $(\min, +)$ branching program does not use this additional feature, it may be helpful for some other minimization problems. Consider, for example, the problem

$$f(x_1, \dots, x_n) = \min\left\{ \sum_{i=1}^{n} a_i x_i \colon \sum_{i=1}^{n} a_i = k \right\}.$$

Since $d(f) = k$ and $w(f) = n$, every (ordinary) $(\min, +)$ branching program for f must have at least kn contacts. But since $f(x) = \min\{kx_1, \dots, kx_n\}$, already n contacts are enough for extended programs. So, it would be interesting to know whether extended $(\min,+)$ branching programs for $W_{n,k}$ must still be of size $\Omega(kn^2)$? Note that Lemma 2 fails for extended $(\min,+)$ branching programs as well. The reason is that then Claim 2 needs not to hold: the number of contacts in a chain may be much smaller than $d(f)$.

References

[1] Bellman, R., 1958. On a routing problem. Quarterly of Appl. Math. 16, 87–90.

[2] Dijkstra, E., 1959. A note on two problems in connection with graphs. Numerische Math. 1, 269–271.

[3] Erdős, P., Gallai, T., 1959. On maximal paths and circuits in graphs. Acta Math. Acad. Sci. Hungar. 10, 337–356.

[4] Floyd, R., 1962. Algorithm 97, shortest path. Comm. ACM 5, 345.

[5] Ford, L., 1956. Network flow theory. Tech. Rep. P-923, The Rand Corp.

[6] Jukna, S., 2015. Lower bounds for tropical circuits and dynamic programs. Theory of Comput. Syst. 57 (1), 160–194,

[7] Kerr, L., 1970. The effect of algebraic structure on the computation complexity of matrix multiplications. Ph.D. thesis, Cornell Univ., Ithaca, N.Y.

[8] Markov, A., 1962. Minimal relay-diode bipoles for monotonic symmetric functions. Problemy Kibernetiki 8, 117–121, english transl. in *Problems of Cybernetics* 8 (1964), 205–212.

[9] Mehlhorn, K., Galil, Z., 1976. Monotone switching circuits and boolean matrix product. Computing 16 (1-2), 99–111.

[10] Moore, E., 1957. The shortest path through a maze. In: Proc. Internat. Sympos. Switching Theory. Vol. II. Harvard Univ. Press 1959, pp. 285–292.

[11] Moore, E., Shannon, C., 1956. Reliable circuits using less reliable relays. J. Franklin Inst. 262 (3), 281–297.

[12] Paterson, M., 1975. Complexity of monotone networks for boolean matrix product. Theoret. Comput. Sci. 1 (1), 13–20.

[13] Potechin, A., 2010. Bounds on monotone switching networks for directed connectivity. In: 51th Ann. IEEE Symp. on Foundations of Comput. Sci., FOCS. pp. 553–562.

[14] Pratt, V., 1975. The power of negative thinking in multiplying boolean matrices. SIAM J. Comput. 4 (3), 326–330.

[15] Schnitger, G., 1983. On depth-reduction and grates. In: Proc. of 24th IEEE Ann. Symp. on Foundations of Comput. Sci. pp. 323–328.

[16] Warshall, S., 1962. A theorem on boolean matrices. J. ACM 9, 11–12.

ECCC ISSN 1433-8092

http://eccc.hpi-web.de