A Lower Bound for the Shortest Path Problem

Ketan Mulmuley
Dept. of Computer Science
University of Chicago
1100 East 58th Street
Chicago, Il 60637, USA
mulmuley@cs.uchicago.edu

Pradyut Shah
Dept. of Computer Science
University of Chicago
1100 East 58th Street
Chicago, Il 60637, USA
pradyut@cs.uchicago.edu

Dept. of Computer Science & Engg.
Indian Institute of Technology, Powai
Mumbai 400 076, India
mulmuley@cse.iitb.ernet.in

Abstract

We show that the Shortest Path Problem cannot be solved in $o(\log n)$ time on an unbounded fan-in PRAM without bit operations using $\text{poly}(n)$ processors even when the bit-lengths of the weights on the edges are restricted to be of size $O(\log^3 n)$. This shows that the matrix-based *repeated squaring* algorithm for the Shortest Path Problem is optimal in the unbounded fan-in PRAM model without bit operations.

1 Introduction

Proving the optimality of algorithms for important combinatorial problems and determining their intrinsic hardness properties requires finding good lower bounds for them. In this paper, we give a lower bound for the Shortest Path Problem in a natural model of computation.

The Shortest Path Problem is the following: given a weighted, directed graph and two special vertices s and t, compute the weight of the shortest path between s and t.

For positive edge weights, Dijkstra's classical algorithm allows us to compute the weight of the shortest path in polynomial time. By comparison, if the graph is permitted to have negative edge weights, then the problem is known to be NP-complete [8].

The model for the lower bound is a variant of the Parallel Random Access Machine (PRAM for short). The PRAM consists of a set of processors that have access to a shared memory. Each processor has a set of registers and local memory and can access the

shared memory at unit cost. The complexity class NC is defined to be the set of problems that can be solved on a PRAM in polylogarithmic time using polynomially many processors.

The Shortest Path Problem is known to be computable in NC by *repeatedly squaring* the adjacency matrix of the graph where the operations $\{+, *\}$ are replaced by $\{\min, +\}$ [14]. On a PRAM this can be done in $O(\log^2 n)$ arithmetic operations with $\text{poly}(n)$ processors.

One of the features of this algorithm is that it is completely arithmetic in nature *i.e.*, it performs arithmetic operations on its inputs, but doesn't look at the individual bits of the inputs. This property is shared by many important parallel algorithms for combinatorial problems. Accordingly, Mulmuley [16] defined a restricted PRAM model without bit operations. Mulmuley's model eliminates those operations that allow bit-extraction or updates of the bits of the individual registers, but provides the usual arithmetic, indirect referencing, conditional and unconditional branch operations at unit cost (independent of the bit-lengths of the operands). We consider here an unbounded fan-in model, in which the operations $\{+, \min, \max\}$ have unbounded fan-in at unit cost (independent of the bit-lengths of the operands). However, multiplication $(*)$ is restricted to have bounded fan-in.

Unlike earlier models used for proving lower bounds, such as the constant-depth [12] or monotone circuit model [19], the PRAM model without bit operations is natural. Virtually all known parallel algorithms for weighted optimization and algebraic problems fit inside the model. Examples include fast parallel algorithms for solving linear systems [6], minimum weight spanning trees [14], shortest paths [14], global min-cuts in weighted, undirected graphs [13], blocking flows and max-flows [9, 21], approximate computation of roots of polynomials [2, 18], sorting algorithms [14] and several problems in computational geometry [20]. In constrast to Boolean circuits where no lower bounds are known for unbounded depth circuits, our result gives a lower bound for a natural problem in a natural model of computation.

There are many natural combinatorial problems that are known to be polynomially computable but have resisted all efforts at efficient parallelization. Many problems such as the WEIGHTED MAX FLOW problem are P–complete [10]. Hence, assuming that P is different from NC, we cannot expect to find fast parallel algorithms for these problems.

In the PRAM model without bit operations, Mulmuley [16] proved a lower bound on the WEIGHTED MAX FLOW problem (or equivalently, the WEIGHTED *s-t* MIN CUT problem). He showed that it is not possible to solve the problem in time $o(n^{1/8})$ using $2^{\Omega(n^{1/8})}$ processors even when the bit-lengths of the weights are $O(n^c)$ for some constant $c > 0$. This rules out the possibility of any fast parallel algorithm (which does not use bit operations) for the problem.

1.1 Technique

The proof of this lower bound begins by giving a lower bound on the *parametric complexity* of the Shortest Path Problem, building on the work of Carstensen [4]. This is a general technique for giving lower bounds on parallel computation times of homoge-

neous weighted combinatorial problems [16].

A weighted combinatorial problem is homogeneous if scaling all the weights in the problem by a factor ρ scales the weight of the answer by ρ as well.

Assume that the weights on the edges of the input graph are not just real numbers, but linear functions in a parameter λ. Then for each value of λ, we can compute the weight of the shortest path in the graph. If we plot the weight of the shortest path as a function of λ, the resulting *optimal cost graph* is piecewise linear and concave. The parametric complexity of the problem for a fixed graph and a fixed set of linear weight functions is defined as the number of *breakpoints i.e.* points at which the function changes slope.

The parametric complexity of the Shortest Path Problem for size n and size parameter β is the maximum parametric complexity over all possible choices of graphs on n vertices and all possible linear weight functions of the form $a+b\lambda$, where the bit-lengths of a and b are restricted to be less than β.

The following theorem (Theorem 3.1.1 from [16]) relates the parametric complexity of a general weighted combinatorial problem to a lower bound on its computation time in the unbounded fan-in PRAM model without bit operations. The proof in Mulmuley's paper only considers the bounded fan-in case but can be extended to the unbounded fan-in case.

Theorem 1.1 (Mulmuley) *Let $\phi(n,\beta(n))$ be the parametric complexity of any homogeneous optimization problem where n denotes the input cardinality and $\beta(n)$ the bit-size of the parameters. Then the decision version of the problem cannot be solved in the PRAM model without bit operations in $o\left(\sqrt{\log \phi(n,\beta(n))}\right)$ time using $2^{\sqrt{\log \phi(n,\beta(n))}}$ processors, even if we restrict every numeric parameter in the input to size $O(\beta(n))$.*

Carstensen [4, 3] proved the following theorem:

Theorem 1.2 (Carstensen) *There is an explicit family of graphs G_n on n vertices with edge weights that are linear functions in a parameter λ, such that the optimal cost graph of the shortest path between s and t has $2^{\Omega(\log^2 n)}$ breakpoints.*

However, Carstensen's proof is very complex and does not take into account the issue of bit-lengths. It is not possible to obtain a lower bound using Theorem 1.1 that is sensitive to bit-lengths without obtaining good bounds on the bit-lengths of the coefficients of the edge weights. We give a simplified proof of her theorem (using a similar construction) which allows us to to take into account the issue of bit-lengths.

Theorem 1.3 *There is an explicit family of graphs G_n on n vertices, with edge weights that are linear functions in a parameter λ, such that the optimal cost graph of the weight of the shortest path between s and t has $2^{\Omega(\log^2 n)}$ breakpoints. In addition, the bit-lengths of the coefficients of the cost functions have size $O(\log^3 n)$. Thus, the parametric complexity of the Shortest Path Problem for graph size n and bit-length $O(\log^3 n)$ is $2^{\Omega(\log^2 n)}$.*

By combining the above result with Theorem 1.1, we get the following theorem:

Theorem 1.4 (Main Theorem) *The Shortest Path Problem cannot be computed in $o(\log n)$ steps on an unbounded fan-in PRAM without bit operations using $\mathrm{poly}(n)$ processors, even if the weights on the edges are restricted to have bit-lengths $O(\log^3 n)$.*

1.2 Tightness

The matrix-based *repeated squaring* algorithm for Shortest Path Problem can be solved in $\varepsilon \log n$ steps for any $\varepsilon > 0$ with $\text{poly}(n)$ processors on a PRAM that allows unbounded fan-in `min` operations (but only bounded fan-in additions), because multiplying k matrices (for any fixed constant k) can be done in 2 steps in this model using n^{k+1} processors.

Since the model for the lower bound assumes unit cost for all operations (including some with unbounded fan-in), the result shows that the above algorithm for the Shortest Path Problem is optimal in the unbounded fan-in PRAM model without bit operations. In particular, the problem cannot be solved using $o(\log n)$ operations in this model using $\text{poly}(n)$ processors even if we restrict the edge weights to have bit-lengths $O(\log^3 n)$ (or, alternatively, magnitude $2^{O(\log^3 n)}$).

1.3 Extensions

There are other combinatorial problems in P, such as WEIGHTED GRAPH MATCHING [15], that have eluded all attempts at efficient parallelization. The problem is not even known to be P–complete. Our motivation behind studying the Shortest Path Problem started by wanting to give a lower bound for the WEIGHTED GRAPH MATCHING problem. Since the Shortest Path Problem on directed graphs can be reduced to WEIGHTED GRAPH MATCHING, our result yields a lower bound for the latter problem as well.

Corollary 1.5 *The* WEIGHTED BIPARTITE MATCHING *problem cannot be solved in* $o(\log n)$ *steps on an unbounded fan-in PRAM without bit operations using* $\text{poly}(n)$ *processors even if the weights on the edges are restricted to have bit-lengths* $O(\log^3 n)$.

We conjecture that it should be possible to obtain super-polylogarithmic lower bounds using similar techniques for WEIGHTED MATCHING in general graphs. It would also be interesting to give similar lower bounds for other problems that are not known to be in NC, nor known to be P–complete. This paper is a step towards that goal.

2 Preliminaries

Definition 2.1 *A directed graph is said to be layered if its vertices can be arranged in columns so that all edges go between vertices in adjacent columns.*

All the graphs used in this paper are directed and layered. We imagine the graph to be embedded in a grid and label each vertex of the graph by its coordinate. The vertex in the r^{th} row and c^{th} column will be labelled as (r, c). Occasionally, we omit the column number for the vertex if it is clear from the context.

Edges are denoted by $(r, c) \to (k, c+1)$ or simply $r \to k$ if the column number is unambiguous. The weights of edges are labelled by $w_{r,k}$. If we wish to emphasize the fact that the weights are linear functions in the parameter λ, we denote the weight as $w_{r,k}(\lambda)$.

All the graphs in this paper will have two special vertices s and t between which we wish to compute the shortest path. Since all the graphs we use are layered, we may assume that the vertex s sits in the 0^{th} column and the vertex t in the last column.

Definition 2.2 *The core of a graph is the graph obtained by eliminating the vertices s and t.*

3 Outline

Theorem 3.1 *For any $m, n \in \mathbb{N}$, there exists a graph with core $G_{m,n}$ having the following properties:*

(i) *$G_{m,n}$ is a layered graph with at most $2^m(2n-1)$ rows and 3^m columns, having exactly n vertices in the first column.*

(ii) *The edges of $G_{m,n}$ are labelled by linear functions in a parameter λ such that the optimal cost graph of the weight of the shortest path between s and t (as a function of λ) has at least n^m breakpoints.*

(iii) *There is an edge from s to each of the n vertices in the first column with weights*

$$w_{s,(i,1)}(\lambda) = \frac{i(i+1)}{2} - i\lambda \qquad (0 \le i < n).$$

(iv) *All the vertices q in the last column of the core are connected to t with weight:*

$$w_{(q,3^m),t} = 0.$$

The substitution $m = \log n$ will yield a graph on $n^{3.585}$ vertices such that the optimal cost graph of the shortest path has at least $2^{\log^2 n}$ breakpoints. We can rephrase this to say that graphs G_n on n vertices can be constructed with $2^{\Omega(\log^2 n)}$ breakpoints on the optimal cost graph of the shortest path. The coefficients involved in this construction will be shown to have bit-lengths $O(\log^3 n)$.

The construction will use negative edge weights, but since the graphs are layered, we can always add a large positive weight to each edge without changing the structure of the optimal paths.

4 Construction

The graph $G_{m,n}$ is constructed inductively from $G_{m-1,n}$. The idea behind the proof is that each optimal path in $G_{m-1,n}$ yields n optimal paths in $G_{m,n}$ with varying slopes, thus increasing the number of breakpoints by a factor of n.

Given a layered graph and a particular shortest path over some fixed interval of the parameter λ, one can easily create n shortest paths by breaking up the interval into n

pieces, appending a new layer of n vertices, and attaching them to the endpoint of the given path with suitable weights. However, the weights would depend on the interval. The goal of the construction is to create a gadget that behaves the same way but with a choice of weight functions that are independent of the interval.

The rest of this paper is laid out as follows. Section 4.1 gives the stronger inductive hypothesis (Lemma 4.1) that we need to prove the theorem. Section 4.2 specifies the intervals. Section 4.3 gives the proof of the main theorem (Thm 3.1). Section 4.4 defines the topology of the graph $G_{m,n}$. Section 4.5 defines the inductive construction of the weights on the edges of the graph $G_{m,n}$. The proof of Lemma 4.1 can be found in section 4.6.

4.1 Inductive Hypothesis

Lemma 4.1 *For any $D_1, D_2 \in \mathbb{R}$, $m, n \in \mathbb{N}$, and $g : \mathbb{Z} \times \mathbb{Z} \to \mathbb{Z}$ such that $g(r, 0) = 0$, there is a graph having core $G_{m,n}$ with the following properties:*

(i) There are n^{m-1} disjoint intervals

$$\mathcal{I}_{j,m} = [\, \alpha_{j,m} + \epsilon, \beta_{j,m} - \epsilon \,] \qquad (0 \le j < n^{m-1})$$

with $\beta_{j,m} - \alpha_{j,m} > n$ and $\epsilon < 1$. The intervals $\mathcal{I}_{j,m}$ will be independent of the parameters D_1, D_2 and g, and will depend only on m and n.

(ii) For each interval, there exist n paths $P_{i,j}$ (from vertices in the first column of G to the last column of G) that are pairwise vertex-disjoint.

Notationally, $P_{i,j}$ always denotes the path in the core of the graph starting from the vertex $(i, 1)$, and $\mathcal{P}_{i,j}$ denotes the s-t path that contains $P_{i,j}$ in it. That is, $\mathcal{P}_{i,j} = (s \to i) \cup P_{i,j} \cup (r_{i,j} \to t)$ where $r_{i,j}$ denotes the last vertex of the path $P_{i,j}$.

(iii) For $0 \le i < n$, $P_{i,j}$ is the optimal path starting from vertex $(i, 1)$ in the interval $\mathcal{I}_{j,m}$.

(iv) For $0 \le i < n$, let $j = nd + r$ where $0 \le r < n$. Then

$$C(P_{i,j})(\lambda) = C(P_{0,j})(\lambda) + iD_1\alpha_{d,m-1} + iD_2\lambda + g(r, i). \tag{1}$$

(v) The difference in cost between $P_{i,j}$ and any other non-optimal path starting at vertex $(i, 1)$ is at least ϵ.

4.2 Construction of the Intervals

Fix $N > mn^3$. Let $j = nd + r$ where $0 \le r < n$. Define $\alpha_{j,m}$ and $\beta_{j,m}$ as follows:

$$\begin{aligned}
\alpha_{0,1} &= 0 \\
\beta_{0,1} &= N^2 \\
\alpha_{j,m} &= \alpha_{nd+r,m} \\
&= N\alpha_{d,m-1} + rN^2 \\
\beta_{j,m} &= \beta_{nd+r,m} \\
&= N\beta_{d,m-1} + (r+1)N^2
\end{aligned}$$

Intuitively, at each stage we stretch the intervals by a factor of N and divide it into n parts. Hence, $\beta_{j,m} - \alpha_{j,m} = N^2 \gg n$, and this satisfies condition (i) of the inductive hypothesis.

4.3 Proof of Theorem 3.1

PROOF: Before we prove Lemma 4.1, it is instructive to see how the main theorem follows from it. Let $G_{m,n}$ be the graph obtained by choosing the parameters $D_1 = N$, $D_2 = 0$ and $g(r,i) = N^2 ir$. Substituting the values into equation (1) and simplifying using the definition of the intervals above, we get the following equation for the optimal paths in the core of the graph:

$$C(P_{i,j})(\lambda) = C(P_{0,j})(\lambda) + i\alpha_{j,m}.$$

Therefore, for s-t paths we have that

$$C(\mathcal{P}_{i,j})(\lambda) = C(\mathcal{P}_{0,j})(\lambda) + i\alpha_{j,m} + \frac{i(i+1)}{2} - i\lambda.$$

Hence, we have that $\mathcal{P}_{i,j}$ is the optimal path for the interval

$$[\alpha_{j,m} + i, \alpha_{j,m} + i + 1] \cap \mathcal{I}_{j,m},$$

and since $\epsilon < 1$, it follows that each of these paths is optimal in a non-zero interval. Each path must have a different slope because of its linear term depends on i. Hence we get n breakpoints in each of the n^{m-1} intervals, yielding n^m breakpoints in all.

□

4.4 Construction of the Core

The graph $G_{m,n}$ is constructed by induction on the parameter m.

The graph $G_{1,n}$ has 3 columns with n, n and $2n-1$ vertices respectively as shown in Figure 1. Each of the n vertices in the first column is connected to the corresponding vertex in the second column, and each vertex $(i,2)$ in the second column is connected by n edges to the vertices $(i+j,3)$ where $0 \le j \le n-1$.

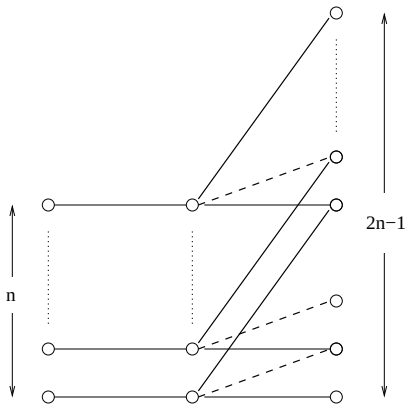

Figure 1: Base Case with Chains

$G_{m,n}$ is constructed recursively from two copies of $G_{m-1,n}$ and a third copy of $G_{m-1,2n-1}$ (which are referred to as G^L, G^M and G^R respectively) as shown in Figure 2.

The first two copies of $G_{m-1,n}$ are connected *back-to-back*. G^M is a reflection of G^L with the edges reversed as well. G^M has exactly n vertices in the last column (because it is a mirror image of G^L). G^R (which is a copy of $G_{m-1,2n-1}$) has $2n-1$ vertices in the first column. We connect the i^{th} vertex in the last column of G^M to the $(i+j)^{\text{th}}$ vertex in the first column of G^R where $0 \leq j \leq n-1$ (similar to the construction in the base case).

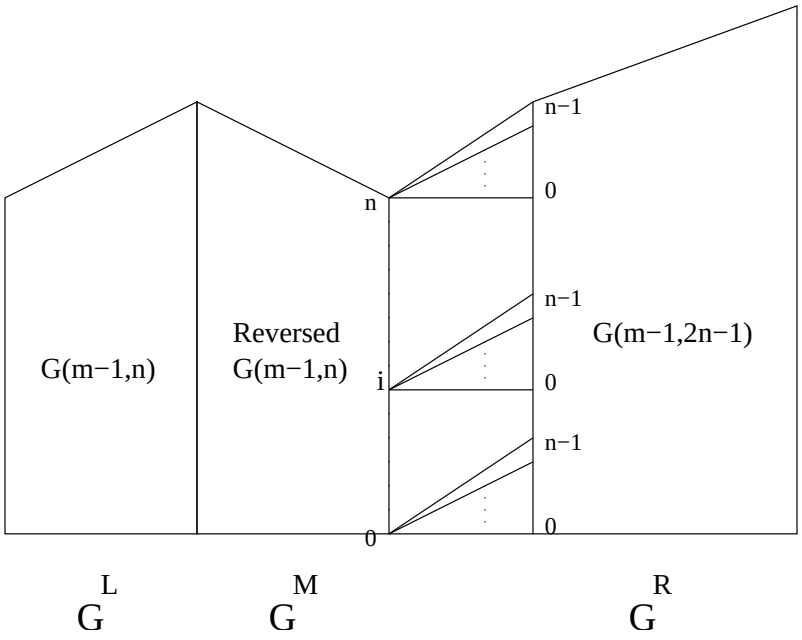

Figure 2: Construction of $G_{m,n}$

4.5 Construction of the Weight Functions

4.5.1 Weight Functions for the Base Case

Fix the parameters D_1, D_2 and the function $g : \mathbb{Z} \times \mathbb{Z} \to \mathbb{Z}$. The parameter K in the definition of the weights will be a constant whose value is fixed later. Define the weights on the edges as follows:

$$\begin{aligned} w_{(k,1),(k,2)} &= 0 \\ w_{(k,2),(k+r,3)} &= \begin{cases} K\left[\frac{r(r+1)}{2}N^2 - r\lambda\right] & k = 0, \\ & 0 \le r < n; \\ w_{(0,2),(r,3)} + kD_1\alpha_{0,1} + kD_2\lambda + g(r,k) & 1 \le k < n, \\ & 0 \le r < n. \end{cases} \end{aligned}$$

4.5.2 Weight Functions for the Inductive Case

Let the parameters to the construction be F_1, F_2 and $h : \mathbb{Z} \times \mathbb{Z} \to \mathbb{Z}$. K_1, K_2 and K_3 are constants whose values will be fixed later.

G^L and G^M are chosen with parameters:

$$\begin{aligned} D_1 &= \frac{N}{2K_3}\left(F_1 - \frac{K_2}{K_1}\right) \\ D_2 &= 0 \\ g(r,i) &= NriD_1. \end{aligned}$$

The graph G^R (which is a copy of $G_{m-1,2n-1}$) is chosen with parameters:

$$\begin{aligned} D_1 &= \frac{N}{K_1} \\ D_2 &= -\frac{1}{K_1} \\ g(r,i) &= \frac{N}{K_1}\frac{i(i+1)}{2} + NriD_1. \end{aligned}$$

Since G^M has exactly n vertices in the last column and G^R has $2n-1$ vertices in the first column, we define the edges analogously to the base case but with the following weights:

$$w_{i,i+r}(\lambda) = h(r,i) - N\frac{K_2}{K_1}\left\{ir + \frac{i(i+1)}{2}\right\} + i\left(F_2 + \frac{K_2}{NK_1}\right)\lambda \qquad (0 \le r < n).$$

The cost functions on the edges of $G_{m,n}$ are defined by shifting the cost functions in G^L and G^M and then scaling them by a factor of K_3, by shifting the cost function in G^R, and then scaling it by a factor of K_2 as follows:

$$w_e(\lambda) = \begin{cases} K_3 \cdot w_e^L\left(\frac{\lambda}{N}\right) & e \in G^L \\ K_3 \cdot w_e^M\left(\frac{\lambda}{N}\right) & e \in G^M \\ K_2 \cdot w_e^R\left(\frac{\lambda}{N}\right) & e \in G^R. \end{cases}$$

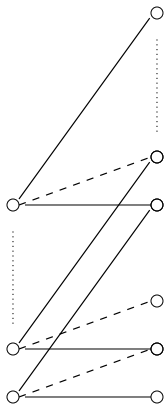

Figure 3: Joining Edges

4.6 Proof of the Inductive Hypothesis

4.6.1 Proof of Lemma 4.1 (Base Case)

PROOF: [BASE CASE]

Define $P_{i,j}$ to be the path $((i,1) \to (i,2)) \cup ((i,2) \to (i+j,3))$. These paths $P_{i,j}$ are vertex-disjoint. All of the conditions of the inductive hypothesis can be easily verified provided:

$$K \gg \frac{\max_{r,i} |g(r,i)|}{\epsilon}.$$

□

4.6.2 Proof of Lemma 4.1 (Inductive Case)

PROOF: [INDUCTIVE CASE]

Fix j and $\lambda \in \mathcal{I}_{j,m}$. Let $j = nd + r$ where $0 \le r < n$. Then $\lambda/N \in \mathcal{I}_{d,m-1}$, and hence the path $P^L_{i,d}$ is optimal in G^L starting from vertex $(i,1)$.

Define $P_{i,j}$ to be $P^L_{i,d} \cup P^M_{i,d} \cup (i \to i+r) \cup P^R_{i+r,d}$. The following two lemmas provide the proof that $P_{i,j}$ is an optimal path starting at vertex $(i,1)$. These paths are vertex-disjoint (as required by the inductive hypothesis).

Lemma 4.2 shows that $P^M_{i,d}$, which is the mirror image of $P^L_{i,d}$, is optimal in G^M. This is not at all clear *a priori* since there is no reason to believe that optimal paths will remain optimal when the edges are reversed. In particular, we would like to have the optimal path in G^M end at $(i, 2 \cdot 3^{m-1})$. Lemma 4.3 finishes up the proof by showing that the path $P^R_{i+r,d}$ is indeed optimal in G^R in the interval $\mathcal{I}_{j,m}$.

Lemma 4.2 *Fix* $\lambda \in \bigcup_{\substack{k=nd+r \\ 0 \le r < n}} \mathcal{I}_{k,m}$. *Then for sufficiently large values of* K_3, *the optimal path in* G^L *and* G^M *starting at node* $(i,1)$ *will be* $P^L_{i,d} \cup P^M_{i,d}$.

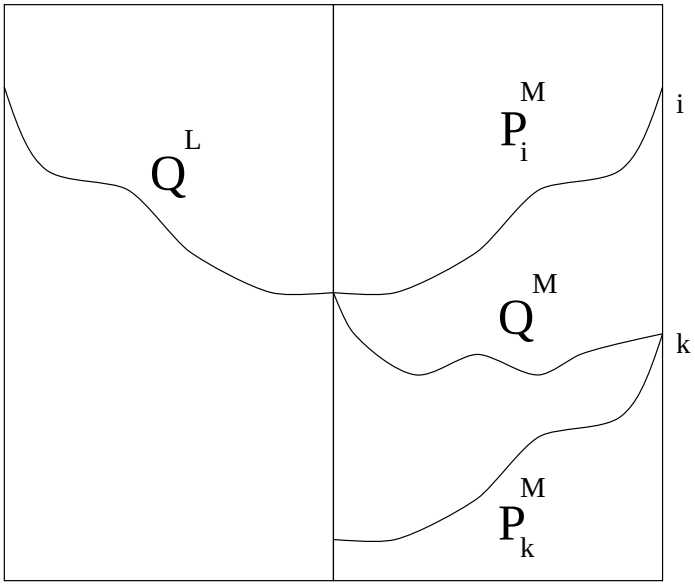

Figure 4: Chains in Lemma 4.2

PROOF: Assume that Q is the optimal path in this interval, and that Q is not symmetric in G^L and G^M. Furthermore, assume without loss of generality that $Q^L = P^L_{i,j}$ and $Q^M \neq P^M_{i,j}$. Let Q^M end at vertex k where $k \neq i$.

The idea of the proof is that the difference between the costs of $P^M_{i,j}$ and $P^M_{k,j}$ is small but the difference in costs between Q^M and $P^M_{k,j}$ is at least ϵ before scaling, and hence at least $K_3\epsilon$ after scaling.

Consider the situation in Figure 4. Before scaling, in the graph $G_{m-1,n}$, the inductive hypothesis guarantees that

$$C(Q^M)(\lambda) - C(P^M_{k,j})(\lambda) > \epsilon.$$

Therefore, after scaling we have that

$$C(Q^M)(\lambda) - C(P^M_{k,j})(\lambda) > K_3\epsilon.$$

The difference in costs between the *parallel* paths $P_{i,j}$ and $P_{k,j}$ is small by the inductive hypothesis:

$$\begin{aligned} \left| C(P^M_{i,j})(\lambda) - C(P^M_{k,j})(\lambda) \right| &\leq nN\left(F_1 - \frac{K_2}{K_1}\right)N^{2+m} \\ &\leq n^{O(m)}\left(F_1 + \frac{K_2}{K_1}\right). \end{aligned}$$

We have that $C(Q)(\lambda) = C(Q^L)(\lambda) + C(Q^M)(\lambda) + C(Q^R)(\lambda)$. Again, without loss of generality Q^R is optimal. It is possible that the path Q gains some advantage in the links between G^M and G^R and also in G^R. If we take the quantity $K_3\epsilon$ to be greater than all

these gains along with the quantity from above, this would contradict the assumption that Q is the optimal path in this interval.

The maximum gain in the intermediate links (from Equation (2)) is

$$\max_{r,i} h(r,i) + 4n^2 N\frac{K_2}{K_1} + n(F_2 + \frac{K_2}{NK_1})N^{1+m}.$$

The maximum gain in G^R is $nN\frac{K_2}{K_1}N^{2+m}$. Clearly $n^{O(m)}\left(F_1 + F_2 + \max|h| + \frac{K_2}{K_1}\right)$ dominates all of the above terms. Thus, choosing K_3 to be

$$K_3 \gg \frac{n^{O(m)}\left(F_1 + F_2 + \max|h| + \frac{K_2}{K_1}\right)}{\epsilon} \tag{2}$$

gives us a contradiction which proves our lemma.

□

Lemma 4.3 *Fix $\lambda \in \bigcup_{\substack{k=nd+r \\ d}} \mathcal{I}_{k,m}$. Then if $\frac{K_2}{K_1}$ is sufficiently large, the optimal path in G^R starting from node $(i,1)$ will be $P^R_{i+r,d}$.*

PROOF: Fix a particular value for d. Since $\lambda \in \mathcal{I}_{nd+r,m}$, therefore $\frac{\lambda}{N} \in \mathcal{I}_{d,m-1}$.

From the previous lemma, it is clear that the optimal paths are symmetric in G^L and G^M and that the optimal path in G^R is $P^R_{k,d}$ for some k. We claim that $k = i + r$.

By adding up the costs, we get that

$$C(P_{i,j})(\lambda) - C(P_{i-1,j})(\lambda) \;=\; \frac{1}{N}\frac{K_2}{K_1}(\alpha_{j,m} - \lambda) + h(r,i) - h(r-1,i)$$

which means that if we impose the condition

$$\frac{K_2}{K_1} \gg \frac{N\max_{r,i}|h(r,i)|}{\epsilon}, \tag{3}$$

then the optimal path will be as required.

□

By the inductive hypothesis, we get the following equations about paths in G^L, G^M, and G^R (before scaling).

$$\begin{aligned}
C(P^L_{i,d})(\lambda) &= C(P^L_{0,d})(\lambda) + \frac{i}{2K_3}\left(F_1 - \frac{K_2}{K_1}\right)\alpha_{d,m-1} \\
C(P^M_{i,d})(\lambda) &= C(P^M_{0,d})(\lambda) + \frac{i}{2K_3}\left(F_1 - \frac{K_2}{K_1}\right)\alpha_{d,m-1} \\
C(P^R_{i,d})(\lambda) &= C(P^R_{0,d})(\lambda) + \frac{1}{K_1}\left\{\frac{Ni(i+1)}{2} + i\alpha_{d,m-1} - i\lambda\right\}.
\end{aligned}$$

After scaling, adding up the costs of the above paths along with the weights of the edges joining the end vertices of G^M to G^R and simplifying, we get:

$$\begin{aligned}
C(P_{i,j})(\lambda) &= C(P^L_{i,d})(\lambda) + C(P^M_{i,d})(\lambda) + w_{i,i+r}(\lambda) + C(P^R_{i+r,d})(\lambda) \\
&= C(P^L_{0,d})(\lambda) + C(P^M_{0,d})(\lambda) + \underbrace{C(P^R_{0,d})(\lambda) + \frac{K_2}{K_1}\left(N\frac{r(r+1)}{2} + r\alpha_{d,m-1} - r\lambda\right)}} \\
&\qquad + iF_1\alpha_{d,m-1} + iF_2\lambda + h(r,i) \\
&= \underbrace{C(P^L_{0,d})(\lambda) + C(P^M_{0,d})(\lambda) + C(P^R_{r,d})(\lambda)} + iF_1\alpha_{d,m-1} + iF_2\lambda + h(r,i) \\
&= C(P_{0,j})(\lambda) + iF_1\alpha_{d,m-1} + iF_2\lambda + h(r,i).
\end{aligned}$$

These yield the required relationships between the optimal paths in the interval $\mathcal{I}_{j,m}$.

Condition (v) is easily satisfied by noting that if we have an optimal and a sub-optimal path starting from vertex $(i,1)$, then it must deviate from the optimal path in either G^L, G^M, G^R, or in the intermediate connecting links. Then the proof of Lemma 4.3 shows that the difference in optimal costs must be at least ϵ. This concludes the proof.

□

5 Analysis

PROOF: [Proof of Theorem 1.3]

From Equations (2) and (3) in section 4.6.2, and the fact that $N = O(n^3 \log n)$ and $\epsilon < 1$, we can rewrite the recurrences for the constants as

$$\begin{aligned}
\frac{K_2}{K_1} &\gg n^{O(1)} \max_{r,i} |h(r,i)| \\
K_3 &\gg n^{O(m)} \left(F_1 + \frac{K_2}{K_1}\right).
\end{aligned}$$

At the topmost level of the recurrence, we choose $F_1 = N$ and $h(r,i) = N^2 ir$, both of which are polynomial in n. However, the function h and parameter F_1 changes as we descend down the construction. In both G^L and G^M we choose $h(r,i) = N^2 ir$, and in G^R we have $|h(r,i)|$ dominated by $\text{poly}(n)\frac{K_2}{K_1}$. We can choose $a > 0$ large enough so that recurrence

$$\left(\frac{K_2}{K_1}\right)_{m-1} \gg \left(\frac{K_2}{K_1}\right)_m n^a$$

has the solution

$$\left(\frac{K_2}{K_1}\right)_r = n^{a(m-r)}.$$

Now in G^L and G^M, the quantity F_1 keeps decreasing by the current value of K_2/K_1, and hence its absolute value increases by at most K_2/K_1. Thus, we can choose $c > 0$

sufficiently large so that

$$(K_3)_r \gg n^{cr} \sum_{r \le t \le m} \left(\frac{K_2}{K_1}\right)_t .$$

This yields the following solution:

$$(K_3)_r = n^{b(m-r)} \qquad \text{for some } b > a > 0.$$

The coefficients grow as the product of the individual multipliers:

$$\begin{aligned} \text{size of coefficients} &= O(n^{b(1+\cdots+(m-1)+m)}) \\ &= O(n^{b\binom{m}{2}}) \\ &= 2^{O(\log^3 n)} \qquad \text{since } m = O(\log n). \end{aligned}$$

Since the magnitude of the coefficients is $2^{O(\log^3 n)}$, it follows that their bit-lengths are $O(\log^3 n)$.

□

Acknowledgements

We would like to thank Dieter van Melkebeek, Marcus Schaefer, Soren Dayton, Varsha Dani and Gina Steele for their help in proof-reading the paper and suggesting valuable comments.

References

[1] Michael Ben-Or. Lower bounds for algebraic computation trees. In 15th *ACM Symposium on the Theory of Computing*, pages 80–86, 1983.

[2] Michael Ben-Or, Ephraim Feig, Dexter Kozen, and Prasoon Tiwari. A fast parallel algorithm for determining all roots of a polynomial with real roots. In *Proceedings of the Eighteenth Annual ACM Symposium on Theory of Computing*, pages 340–349, Berkeley, California, 28–30 May 1986.

[3] Patricia Carstensen. Complexity of some parametric integer and network programming problems. In *Mathematical Programming*, volume 26, pages 64–75, 1983.

[4] Patricia Carstensen. *The Complexity of Some Problems in Parametric Linear and Combinatorial Programming*. PhD thesis, Univ. of Michigan, 1983.

[5] Patricia Carstensen. Parametric cost shortest chain problem. published in ORSA/TIMS, 1987.

[6] L. Csanky. Fast parallel matrix inversion algorithms. *SIAM Journal on Computing*, 5(4):618–623, December 1976.

[7] A. Frank and Éva Tardos. An application of simultaneous Diophantine approximation in combinatorial optimization. *Combinatorica*, 7:49–65, 1987.

[8] Michael R. Garey and David S. Johnson. *Computers and Intractability : A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, 1979.

[9] Andrew V. Goldberg and Robert E. Tarjan. A new approach to the maximum flow problem. In *Proceedings of the Eighteenth Annual ACM Symposium on Theory of Computing*, pages 136–146, Berkeley, California, 28–30 May 1986.

[10] Raymond Greenlaw, H. James Hoover, and Walter L. Ruzzo. *Limits to Parallel Computation : P-Completeness Theory*. Oxford Univ. Press, 1995.

[11] Martin Grötschel, László Lovász, and Alexander Schrijver. *Geometric Algorithms and Combinatorial Optimization*. Springer, Berlin, 1988.

[12] Johan Håstad. Almost optimal lower bounds for small depth circuits. In *Proceedings of the Eighteenth Annual ACM Symposium on Theory of Computing*, pages 6–20, Berkeley, California, 28–30 May 1986.

[13] David Karger and Rajeev Motwani. An NC algorithm for minimum cuts. *SIAM Journal on Computing*, 26, 1997.

[14] F. Thomson Leighton. *Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes*. Morgan Kaufmann, San Mateo, 1992.

[15] László Lovász and M. D. Plummer. *Matching Theory*, volume 29 of *Annals of Discrete Mathematics*. North-Holland Math. Studies, 1986.

[16] Ketan Mulmuley. Lower bounds in a parallel model without bit operations. *SIAM Journal on Computing*, 28, 1998.

[17] Ketan Mulmuley, Umesh V. Vazirani, and Vijay V. Vazirani. Matching is as easy as matrix inversion. *Combinatorica*, 7:105–113, 1987.

[18] C. Andrew Neff. Specified precision polynomial root isolation is in NC. *Journal of Computer and System Sciences*, 48(3):429–463, June 1994.

[19] Alexander Razborov. Lower bounds on the complexity of some boolean functions. *Dokl. Ak. Nauk.*, 1985.

[20] John H. Reif. *Synthesis of Parallel Algorithms*. Morgan Kaufmann, San Mateo, 1993.

[21] Yossi Shiloach and Uzi Vishkin. An $O(n^2 \log n)$ parallel MAX-FLOW algorithm. *Journal of Algorithms*, 3(2):128–146, June 1982.

[22] Andrew Yao. Lower bounds for algebraic computation trees with integer inputs. In *Proceedings of the* 30^{th} *Annual IEEE Symposium on Foundations of Computer Science*, pages 222–227, 1977.