Project Number 288094

eCOMPASS

eCO-friendly urban Multi-modal route PlANning Services for mobile uSers

STREP
Funded by EC, INFSO-G4(ICT for Transport) under FP7

eCOMPASS – TR – 016

Approximate counting of approximate solutions

Mat Mihalk, Rastislav rmek, and Peter Widmayer

December 2016

Approximate counting of approximate solutions (WIP)

Matúš Mihalák, Rastislav Šrámek, and Peter Widmayer

Department of Computer Science, ETH Zurich, Zurich, Switzerland,
e-mail: {mmihalak,rsramek,widmayer}@inf.ethz.ch

Abstract. We present a fully polynomial-time approximation scheme for counting paths shorter than some threshold on a directed, acyclic graph.

1 Introduction

For a long time, people have been interested in counting solutions of combinatorial optimization problems. Most of the past work was focused on either calculating the number of optimal solutions of a problem, for instance the number of shortest s–t paths, or the total number of possible solutions, for instance the number of perfect matchings in a graph. Less thought has been given to the question of counting the number of approximate solutions of a problem, that is, solutions that have cost at most ρ times larger than the cost of the optimal solution, for some constant ρ. We will look at the latter problem in the setting of three combinatorial optimization problems.

Our primary motivation stems from our previous work [1], in which we introduced a problem-based measure of instance similarity and a method for predicting robust solutions of an optimization problem. In it we suppose that we are given an optimization problem that admits several, not necessarily optimal, "feasible" solutions. If the problem is, for instance, to find a shortest s–t path in a concrete graph, the feasible solutions would be all the paths from s to t. In order to be able to find a robust solutions to the optimization problem, we have to be able to determine, for a given ρ, the number of feasible solutions that ρ-approximate the optimal solution. That is, they have cost that is at most ρ times higher than the cost of the optimal solution. Additionally, when given two instances with identical sets of feasible solutions, we want to find the number of solutions that ρ-approximate both instances at the same time.

Counting solutions that approximate the optimum within some factor is related to a number of previously studied problems. The most immediate

one is the enumeration of problem solutions in the order of increasing cost. For instance, a number of previous works deals with enumerating s-t paths in a graph by cost [], focusing on the minimization of the time and space needed to produce the next solution in the procedure. In many cases, however, the number of solutions to a problem is exponential and these enumeration techniques are too slow to be useful for us. We want to count without explicitly listing each solution.

We will briefly introduce the three combinatorial optimization problems which we want to study in this context.

1.1 Counting small sums in $X_1 + X_2 + \ldots + X_n$

Let $X_1,\ldots,X_n$ be sets of integers. Given a value S, we are interested in the number of ways one can choose $(x_1, x_2, \ldots, x_n) \in X_1 \times X_2 \times \ldots \times X_n$ such that $\sum_i x_i < S$. This is a variant of a problem that was introduced by Mizoguchi and Johnson [2]. They asked for the element with K-th smallest sum from $X_1 + X_2 + \ldots + X_n$ and showed that this problem is NP-complete.

Having an efficient algorithm would allow us to combine pre-computed parts of different optimization problems. For instance, consider a graph with two articulations a_1 and a_2. If s and t are such that any s-t path must go through both a_1 and a_2, one can pre-compute lengths of all s-a_1 paths, a_1-a_2 paths and a_2-t paths, consider the pre-computed values as sets X_1, X_2, X_3, and get the resulting approximation set sizes by solving the $X_1 + X_2 + \ldots + X_n$ problem.

TODO:Look around for papers where people use it.

1.2 Shortest paths on DAGs

The second problem we look at is the shortest s-t path problem, where the underlying graph is directed and acyclic. The problem has a large number of applications in many areas, the ones that specifically interest us arise in the context of analysis of biological data. For instance de novo peptide sequencing [3], [4], or sequence alignment [5]. Naor and Brutlag [5] have asked how one can efficiently count sub-optimal paths in order to learn more about DNA sequence alignments. Our work should be a comprehensive answer to this question.

1.3 NP-completeness reduction

We show the unsurprising fact that counting the number of approximate solutions for all three problems is NP-Complete. In 1978, Johnson and

Mizoguchi showed that finding the K-th smallest of the sums $\sum_i^n x_i$ when for all i, $x_i \in X_i$, is NP-complete for n larger than 2. Reduction to this problem does not seem straight-forward, since it requires identifying the concrete elements $x_1, \ldots, x_n$, but we can do a similar reduction to the partition problem.

Consider the decision version of the partition problem: for a given set of positive integers $S = s_1, \ldots, s_n$, is there a partition of S into sets S_1 and S_2 such that the sums of integers in both sets are equal? We will show that if we can count the number of selections $x_i \in X_i$ such that $x_1 + x_2 + \ldots + x_n < S$ for any S, we can solve the decision version of the subset sum problem.

Let $X_i = \{-s_i, s_i\}$. The corresponding subset sum problem has a solution, if we can achieve a choice of $x_i \in X_i$ such that the sum of x_i is equal to 0. If we can efficiently count sums in $X_1 + X_2 + \ldots + X_n$, we can count the number of solutions when the value of S is 0 and 1. If these two counts are not equal, there must exist a solution with the sum of 0 and the answer to the partition problem is "yes". Otherwise the answer is "no". Since the partition problem is known to be weakly NP-Complete [], the $X_1 + X_2 + \ldots + X_n$ problem is weakly NP-hard.

1.4 Reduction for shortest paths

We will consider a graph with $n+1$ vertices, a_1 to a_{n+1}. For each element $x_i \in X_j$ we will add an edge between vertices a_j and a_{j+1} with weight x_i. If we denote $s := a_1$ and $t := a_{n+1}$, the number of $X_1 + X_2 + \ldots + X_n$ sums within a factor ρ of the smallest sum is equal to the number of s to t paths shorter than ρ times the length of the shortest path. Note that the existence of parallel edges is not necessary for the reduction, we could bisect each parallel edge creating a auxiliary vertex to form a graph of equivalent function but without parallel edges.

2 Directed acyclic graphs and X+Y sets

In this section we present fully polynomial time approximation schemes for the problem of counting approximate shortest paths in a directed acyclic graph and for the problem of counting small sums in $X_1 + X_2 + \ldots + X_n$. We will start with shortest paths and first treat the case where we are interested only in the set of approximate solutions for one instance.

2.1 FPTAS for directed acyclic graphs

We first show a recurrence that can be used to exactly count the number of paths that approximate the shortest path within some multiplicative threshold ρ. Evaluating the recurrence takes exponential time, but we will later show how to group partial solutions together in such way that we trade accuracy for the number of recursive calls. We adapt the approach of Stefankovic et al. [6], which they used to approximate the number of all feasible solutions to the knapsack problem.

Let G be a directed acyclic graph with n vertices. We will label the vertices $v_1, \ldots, v_n$ in such order that there is no path from v_i to v_j unless $i < j$, i.e. $v_1, \ldots, v_n$ defines a topological ordering. We suppose that $v_1 = s$ and $v_n = t$, otherwise the graph can be pruned by discarding all vertices that appear before s and after t in the topological order, since no path from s to t ever visits these.

For a concrete vertex v_i with in-degree d, let us denote its d neighbors that precede it in the topological order by $p_1, \ldots, p_d$ and let us denote the corresponding incoming edge lengths by $l_1, \ldots, l_d$. Instead of asking for the number of s-t paths that are shorter than L for a given L, we indirectly ask for smallest threshold L, such that there are at least a paths from s to t, shorter than L. Let $\tau(v_i, a)$ denote the minimum length L such that there are at least a paths from v_1 to v_i of length at most L. $\tau(v_i, a)$ can be computed by the recurrence

$$\begin{aligned}
&\tau(v_1, 0) = -\infty \\
&\tau(v_1, a) = 0, \forall a : 0 < a \leq 1 \\
&\tau(v_1, a) = \infty, \forall a : a > 1 \\
&\tau(v_i, a) = \min_{\substack{\alpha_1,\ldots,\alpha_d \\ \sum \alpha_j = 1}} \max \begin{cases} \tau(p_1, \alpha_1 a) + l_1 \\ \vdots \\ \tau(p_d, \alpha_d a) + l_d \end{cases}.
\end{aligned}$$

Intuitively, the at least a paths starting at v_1 and arriving at v_i must split in some way among incoming edges. The values α_j define such split. We look for a set of $\alpha_1, \ldots, \alpha_d$ that minimizes the maximum allowed path length needed such that the incoming paths can be distributed according to α_j, $j = 1, \ldots, d$.

To find the number of paths of length at most L, we search for a such that $\tau(v_n, a) \leq L < \tau(v_n, a+1)$. In particular, if the length of the shortest s-t path is OPT, we can find the number of ρ-approximate s-t paths by setting $L := \rho OPT$.

Calculating τ using the given recurrence will not result in a polynomial time algorithm since we might need to consider an exponential number of values for a, namely 2^{n-2} on a DAG with maximal number of edges. To overcome this, we will consider only a polynomial number of possible values for a, and always round down to the closest one in the evaluation. If we are looking for an algorithm that counts with $1+\varepsilon$ precision, the ratio between two successive considered values of a must be at most $1+\varepsilon$.

For this purpose, we introduce a new function τ'. In order to achieve precision of $1+\varepsilon$, we will only consider values of τ' for minimum path numbers in the form of q^k for all positive integers k such that $q^k < 2^{n-2}$, where $q = \sqrt[n+1]{1+\varepsilon}$. The values of τ' for other numbers of paths will be undefined. The function τ' is defined by the following recurrence.

$$\begin{aligned}
&\tau'(v_1, 0) = -\infty \\
&\tau'(v_1, a) = 0, \forall a : 0 < a \leq 1 \\
&\tau'(v_1, a) = \infty, \forall a : a > 1 \\
&\tau'(v_i, q^j) = \min_{\substack{\alpha_1,\ldots,\alpha_d \\ \sum \alpha_j = 1}} \max \begin{cases} \tau'(p_1, q^{\lfloor j + \log_q \alpha_1 \rfloor}) + l_1 \\ \vdots \\ \tau'(p_i^d, q^{\lfloor j + \log_q \alpha_d \rfloor}) + l_d \end{cases}
\end{aligned} \tag{1}$$

To give a meaning to the expression $q^{\lfloor j+\log_q \alpha_i \rfloor}$ when $\alpha_i = 0$, we define it, for our purposes, to be equal to 0, which is consistent with its limit when α_i goes to 0. We want to show that the rounding does not make the values of τ' too different from the values of τ. Ideally, we would want the bound $\tau(v_i, q^{j-1}) \leq \tau'(v_i, q^j) \leq \tau(v_i, q^j)$, but we will see that it is sufficient to show that $\tau(v_i, q^{j-i}) \leq \tau'(v_i, q^j) \leq \tau(v_i, q^j)$.

Lemma 1. *Let $1 \leq i$ and $i \leq j$. Then,*

$$\tau(v_i, q^{j-i}) \leq \tau'(v_i, q^j) \leq \tau(v_i, q^j). \tag{2}$$

Proof. We first prove the first inequality, proceeding by induction on i. The base case holds since $\tau(v_1, a) \leq \tau'(v_1, b)$ for any $a \leq b$. Suppose now that the first inequality of (2) holds for every p, $p < i$. Then, for every $0 \leq \alpha < 1$,

$$\begin{aligned}
&\tau'(p, q^{\lfloor j+\log_q \alpha \rfloor}) \geq \tau(p, q^{\lfloor j+\log_q \alpha \rfloor - p}) \\
&\geq \tau(p, q^{j-p-1+\log_q \alpha}) \geq \tau(p, \alpha q^{j-i})
\end{aligned}$$

Thus, since every predecessor of v_i is earlier in the vertex ordering, we can use the obtained inequality to get the bound:

$$\begin{aligned}
\tau'(v_i, q^j) &= \min_{\substack{\alpha_1,\ldots,\alpha_d \\ \sum \alpha_j = 1}} \max \begin{cases} \tau'(p_1, q^{\lfloor j + \log_q \alpha_1 \rfloor}) + l_1 \\ \vdots \\ \tau'(p_d, q^{\lfloor j + \log_q \alpha_d \rfloor}) + l_d \end{cases} \\
&\geq \min_{\substack{\alpha_1,\ldots,\alpha_d \\ \sum \alpha_j = 1}} \max \begin{cases} \tau(p_1, \alpha_1 q^{j-i}) + l_1 \\ \vdots \\ \tau(p_d, \alpha_d q^{j-i}) + l_d \end{cases} \\
&= \tau(v_i, q^{j-i})
\end{aligned}$$

The inequality $\tau'(v_i, q^j) \leq \tau(v_i, q^j)$ follows by a simpler induction on i. Base case holds since $\tau(v_1, x) = \tau'(v_1, x)$ for all x. Assume that the second part of (2) holds for all $p < i$. Then,

$$\tau'(p, q^{\lfloor j + \log_q \alpha_i \rfloor}) \leq \tau(p, q^{\lfloor j + \log_q \alpha_i \rfloor}) \leq \tau(p, \alpha_i q^j).$$

And we can use the recursive definition to obtain the claimed inequality $\tau'(v_i, q^j) \leq \tau(v_i, q^j)$.

$$\begin{aligned}
\tau'(v_i, q^j) &= \min_{\substack{\alpha_1,\ldots,\alpha_d \\ \sum \alpha_j = 1}} \max \begin{cases} \tau'(p_1, q^{\lfloor j + \log_q \alpha_1 \rfloor}) + l_1 \\ \vdots \\ \tau'(p_d, q^{\lfloor j + \log_q \alpha_{d_i} \rfloor}) + l_d \end{cases} \\
&\leq \min_{\substack{\alpha_1,\ldots,\alpha_d \\ \sum \alpha_j = 1}} \max \begin{cases} \tau(p_1, \alpha_1 q^j) + l_1 \\ \vdots \\ \tau(p_d, \alpha_{d_i} q^j) + l_d \end{cases} \\
&= \tau(v_i, q^j)
\end{aligned}$$

□

The next theorem shows how to use τ' to produce a $(1+\varepsilon)$-approximation for the counting problem. We need to show that for any L, if we find k such that $\tau'(v_n, q^k) \leq L < \tau'(v_n, q^{k+1})$, the value q^k will be no more than $(1+\varepsilon)^{\pm 1}$ away from the value a for which $\tau(v_n, a) \leq L < \tau(v_n, a)$.

Theorem 1. *Given L, let k be such that $\tau'(v_n, q^k) \leq L < \tau'(v_n, q^{k+1})$ and a be such that $\tau(v_n, a) \leq L < \tau(v_n, a+1)$. Then $(1+\varepsilon)^{-1} \leq \frac{a}{q^k} \leq 1+\varepsilon$.*

Proof. Applying Lemma 1 twice, we get $\tau(v_n, q^{k-n}) \leq \tau'(v_n, q^k) \leq L < \tau'(v_n, q^{k+1}) \leq \tau(v_n, q^{k+1})$. As $\tau(v_n, q^{k-n})$ is at most L, a is largest such that $\tau(v_n, a) \leq L$, and τ is monotonous in its second parameter, it must be that $q^{k-n} \leq a$. Similarly, $\tau(v_n, q^{k+1})$ is larger than L, so by monotonicity $a \leq q^{k+1}$. Thus both a and q^k must lie between q^{k-n} and q^{k+1} and their ratio can be at most $q^{k+1-(k-n)} = q^{n+1} = 1 + \varepsilon$ and at least $q^{k-(k+1)} = (1+\varepsilon)^{-1/(n+1)} > (1+\varepsilon)^{-1}$. □

Left is to show that the we can compute all values of the function τ' in polynomial time.

Theorem 2. *Given maximum allowed path length L, we can find k which satisfies $\tau'(v_n, q^k) \leq L < \tau'(v_n, q^{k+1})$ in time $O(mn^2\varepsilon^{-1} \log n)$, where m denotes the number of edges in the graph.*

Proof. As noted before, directed acyclic graph on n vertices has at most 2^{n-2} paths between any two vertices. The values of a in τ therefore span at most $\{1, 2, \ldots, 2^{n-2}\}$, and the values of q^k in τ' span at most $\{1, q, q^2, \ldots, q^s\}$, where

$$s := \log_q(2^{n-2}) = \frac{(n-2)}{\log_2 q} = \frac{(n-2)(n+1)}{\log_2(1+\varepsilon)} = O(n^2\varepsilon^{-1}).$$

Thus, we evaluate the function τ' for at most $ns = O(n^3\varepsilon^{-1})$ different parameter pairs.

To show that the evaluation of τ' can be done in polynomial time, we need to show that we can efficiently find $\alpha_1, \ldots, \alpha_d$ that minimize Expression (1). Fortunately, $\tau'(v_i, q^k)$ is monotonous with increasing k, we can thus apply a greedy approach. Given v_i, we will evaluate $\tau'(v_i, q^k)$ for all possible values of q^k in one run. Instead of the tuple $\alpha_1 \ldots \alpha_d$ we will consider an integer tuple $k_1 \ldots k_d$. We start with all k_i equal to 0 and always increase the k_i that minimizes $\tau'(p_i, q^{k_i+1}) + l_i$ by one. Whenever the sum of all q_i^k increases over some value q^k, we store the current maximum of $\tau'(p_i, q^{k_i}) + l_i$ as the value $\tau'(v_i, q^k)$. We terminate once $\sum_i q^{k_i}$ reaches 2^{n-2}. As we can increase each k_i at most s times, we make at most ds steps, each of which involves choosing a minimum from d values and replacing it with a new value. The latter can be done in time $O(\log d) \subseteq O(\log n)$, for instance by keeping the values $\tau'(v_i, q^{k_i+1}) + l_i$ in a heap. The sum of d for all vertices is equal to the number of edges m. The time necessary to find all the values of $\tau'(v_n, q^j)$ is thus $O(mn^2\varepsilon^{-1} \log n)$. □

2.2 FPTAS for solutions that approximate two instances

In this section we consider counting solutions that are ρ-approximate for two instances at the same time. The instances differ in edge lengths, but share the same topology,[1] effectively giving each edge two different lengths. We cannot directly apply the approach for the single instance case as we now have two lengths per edge and it is unclear how to define a maximum over pairs in Equation (1). We will nonetheless follow a similar approach and define a function τ_2 similar to τ that adds one of the edge lengths in a form of a "budget". $\tau_2(v_i, a, L_1)$ will be equal to the shortest length L_2 with respect to the edge lengths in the second instance such that there are at least a paths from v_1 to v_i, no longer than L_1 with respect to the edge lengths in the first instance. We will denote the edge lengths of the d incoming edges of vertex v_i in the second instance by l'_i. τ_2 can then be evaluated by the following recursion.

$$
\begin{aligned}
&\tau_2(v_1, 0, x) = -\infty, \forall x \in \mathbb{R}^+ \\
&\tau_2(v_1, a, x) = 0, \forall a : 0 < a \leq 1, \forall x \in \mathbb{R}^+ \\
&\tau_2(v_1, a, x) = \infty, \forall a : a > 1, \forall x \in \mathbb{R}^+ \\
&\tau_2(v_i, a, L_1) = \min_{\substack{\alpha_1,\ldots,\alpha_d \\ \sum \alpha_j = 1}} \max \begin{cases} \tau_2(p_1, \alpha_1 a, L_1 - l_1) + l'_1 \\ \vdots \\ \tau_2(p_d, \alpha_d a, L_1 - l_d) + l'_d \end{cases}
\end{aligned}
$$

If used to solve the problem, the function τ_2 would have to be evaluated not only for an exponential number of path counts a but also for possibly exponential number of values of L_1. To end up with polynomial runtime, we need to consider only a polynomial number of values for both. We will introduce a function τ'_2 that does this by considering only path lengths in the form of r^k, where $r = \sqrt[n]{1+\delta}$, and path numbers a in the form of q^j,

[1] Another way of looking at this is that if we have a bijection that tells us which vertex in the first instance corresponds to which vertex in the second instance, we can drop all edges that exist in only one of them, as any path using this will not approximate the optimal solution in the other instance.

where $q = \sqrt[n]{1+\varepsilon}$, for positive ε and δ.

$$\tau_2'(v_1, 0, x) = -\infty, \forall x \in \mathbb{R}^+$$
$$\tau_2'(v_1, a, x) = 0, \forall a : 0 < a \leq 1, \forall x \in \mathbb{R}^+$$
$$\tau_2'(v_1, a, x) = \infty, \forall a : a > 1, \forall x \in \mathbb{R}^+$$
$$\tau_2'(v_i, q^j, r^k) = \min_{\substack{\alpha_1,\ldots,\alpha_d \\ \sum \alpha_j = 1}} \max \begin{cases} \tau_2'(p_1, q^{\lfloor j + \log_q \alpha_1 \rfloor}, r^{\lfloor \log_r (r^k - l_1) \rfloor}) + l_1' \\ \vdots \\ \tau_2'(p_d, q^{\lfloor j + \log_q \alpha_d \rfloor}, r^{\lfloor \log_r (r^k - l_d) \rfloor}) + l_d' \end{cases}$$

We again show that τ_2' approximates τ_2, this time in two variables.

Lemma 2. *Let $0 \leq i$, $i \leq j$, and $i \leq k$. Then*

$$\tau_2(v_i, q^{j-i}, r^k) \leq \tau_2'(v_i, q^j, r^k) \leq \tau_2(v_i, q^j, r^{k-i}). \tag{3}$$

Proof. We proceed as for Lemma 1. Note that the function τ_2 is monotone non-decreasing in a, but monotone non-increasing in L_1. Proceeding by induction on i, the base case again holds since $\tau_2(v_1, a, y) \leq \tau_2'(v_1, b, y)$ for any $a \leq b$ and y. We suppose that Equation (3) holds for all $p < i$. Then, for every $0 \leq \alpha < 1$,

$$\tau_2'(p, q^{\lfloor j + \log_q \alpha \rfloor}, r^{\lfloor \log_r (r^k - l) \rfloor}) \geq \tau_2(p, q^{\lfloor j + \log_q \alpha \rfloor - p}, r^{\lfloor \log_r (r^k - l) \rfloor})$$
$$\geq \tau_2(p, q^{j-p-1+\log_q \alpha}, r^k - l) \geq \tau_2(p, \alpha q^{j-i}, r^k - l).$$

Thus, since every predecessor of v_i has index smaller than i,

$$\begin{aligned} \tau_2'(v_i, q^j, r^k) &= \min_{\substack{\alpha_1,\ldots,\alpha_d \\ \sum \alpha_j = 1}} \max \begin{cases} \tau_2'(p_1, q^{\lfloor j + \log_q \alpha_1 \rfloor}, r^{\lfloor \log_r (r^k - l_1) \rfloor}) + l_1' \\ \vdots \\ \tau_2'(p_d, q^{\lfloor j + \log_q \alpha_d \rfloor}, r^{\lfloor \log_r (r^k - l_d) \rfloor}) + l_d' \end{cases} \\ &\geq \min_{\substack{\alpha_1,\ldots,\alpha_d \\ \sum \alpha_j = 1}} \max \begin{cases} \tau_2(p_1, \alpha_1 q^{j-i}, r^k - l_1) + l_1' \\ \vdots \\ \tau_2(p_d, \alpha_d q^{j-i}, r^k - l_d) + l_d' \end{cases} \\ &= \tau_2(v_i, q^{j-i}, r^k). \end{aligned}$$

The proof of the inequality $\tau_2'(v_i, q^j, r^k) \leq \tau_2(v_i, q^j, r^{k-i})$ is similar. Assuming that (3) holds for every $p < i$, we obtain

$$\tau_2'(p, q^{\lfloor j + \log_q \alpha \rfloor}, r^{\lfloor \log_r (r^k - l) \rfloor}) \leq \tau_2(p, q^{\lfloor j + \log_q \alpha \rfloor}, r^{\lfloor \log_r (r^k - l) \rfloor - p})$$
$$\leq \tau_2(p, \alpha q^j, r^{\log_r (r^k - l) - p - 1}) \leq \tau_2(p, \alpha q^j, r^{k-i} - l).$$

Plugging it into the definition of τ_2', we obtain

$$\begin{aligned}\tau_2'(v_i, q^j, r^k) &= \min_{\substack{\alpha_1,\ldots,\alpha_d \\ \sum \alpha_j = 1}} \max \begin{cases} \tau_2'(p_1, q^{\lfloor j + \log_q \alpha_1 \rfloor}, r^{\lfloor \log_r (r^k - l_1) \rfloor}) + l_1' \\ \vdots \\ \tau_2'(p_d, q^{\lfloor j + \log_q \alpha_d \rfloor}, r^{\lfloor \log_r (r^k - l_d) \rfloor}) + l_d' \end{cases} \\ &\leq \min_{\substack{\alpha_1,\ldots,\alpha_d \\ \sum \alpha_j = 1}} \max \begin{cases} \tau_2(p_1, \alpha_1 q^j, r^{k-1} - l_1) + l_1' \\ \vdots \\ \tau_2(p_d, \alpha_d q^j, r^{k-1} - l_d) + l_d' \end{cases} \\ &= \tau_2(v_i, q^j, r^{k-1}).\end{aligned}$$

□

Using Lemma 2, we can show that τ_2' provides enough information to compute an approximation of τ_2. However, we cannot get a $(1+\varepsilon)$ approximation to the optimal value as in the Theorem 1, because we need to round the value of L_1 to a power of r in order for it to be legal parameter of τ_2' and we further round it during the evaluation of τ_2'. We will therefore relate the result of τ_2' to the results of τ_2 we would have gotten if we considered the value of L_1 when rounded up towards the nearest number that can be represented as r^k for integer k and the value r^{k-n}. Due to the choice of r, the ratio of these two values is $1+\delta$.

Theorem 3. *Given a desired maximum v_1-v_n path lengths L_1 and L_2 with respect to two instances, let integer k be such that $\tau_2'(v_n, q^k, r^{\lceil \log_r L_1 \rceil}) \leq L_2 < \tau_2'(v_n, q^{k+1}, r^{\lceil \log_r L_1 \rceil})$, a be such that $\tau_2(v_n, a, r^{\lceil \log_r L_1 \rceil - n}) \leq L_2 < \tau_2(v_n, a+1, r^{\lceil \log_r L_1 \rceil - n})$, and b be largest such that $\tau_2(v_n, b, r^{\lceil \log_r L_1 \rceil}) \leq L_2 < \tau_2(v_n, a+1, r^{\lceil \log_r L_1 \rceil})$. Then $a \leq b$, $\frac{a}{q^k} \leq 1+\varepsilon$ and $\frac{q^k}{b} \leq 1+\varepsilon$.*

Proof. The statement that $a \leq b$ follows from the definition of a and b: decreasing the limit on the path length in the first instance from $r^{\lceil \log_r L_1 \rceil}$ to $r^{\lceil \log_r L_1 \rceil - n}$ cannot increase the number of possible paths. By applying Lemma 2 twice, we get

$$\tau_2(v_n, q^{k-n}, r^{\lceil \log_r L_1 \rceil}) \leq \tau_2'(v_n, q^k, r^{\lceil \log_r L_1 \rceil}) \leq L_2, \tag{4}$$

and

$$L_2 < \tau_2'(v_n, q^{k+1}, r^{\lceil \log_r L_1 \rceil}) \leq \tau_2(v_n, q^{k+1}, r^{\lceil \log_r L_1 \rceil - n}). \tag{5}$$

From the definition of a and (5) we can conclude $a \leq q^{k+1}$. This implies that $\frac{a}{q^k} \leq q \leq 1+\varepsilon$, due to our choice of q. Similarly, from the definition of b and (4) we get $b \geq q^{k-n}$ and thus $\frac{q^k}{b} \leq q^n \leq 1+\varepsilon$. □

Theorem 3 shows that the computed number of s-t paths q^k cannot be larger than b by more than a factor of $1+\varepsilon$, nor can it be smaller than a by a factor larger than $1+\varepsilon$. We can now state the overall running time of the approach. Compared to the function τ' we need to evaluate τ_2' for $\lceil \log_r L_1 \rceil = O(n\delta^{-1} \log L_1)$ values of r^l, in addition to the values of v_i and q^k. Otherwise the arguments are identical to the proof of Theorem 2. Note that $\log L_1$ is by definition in $O(n)$, but we list it explicitly since it can be much smaller in practice.

Theorem 4. *Given maximum allowed path lengths L_1 and L_2 in two instances, we can find k which satisfies $\tau_2'(v_n, q^k, r^{\lceil \log_r L_1 \rceil}) \leq L < \tau_2'(v_n, q^{k+1}, r^{\lceil \log_r L_1 \rceil})$ in time $O(mn^3\varepsilon^{-1}\delta^{-1} \log n \log L_1)$, where m denotes the number of edges in the graph.*

2.3 Pseudo-polynomial algorithm for two instances

If the discrepancy between a and b as defined in the Theorem 3 is too large and all edges have integer lengths, we can consider all possible lengths in the first instance, instead of rounding to values in the form of r^k. The function τ_2'' will be τ' extended with exact maximum length of path in the first instance.

$$
\begin{aligned}
&\tau_2''(v_1, 0, x) = -\infty, \forall x \in \mathbb{R}^+ \\
&\tau_2''(v_1, a, x) = 0, \forall a : 0 < a \leq 1, \forall x \in \mathbb{R}^+ \\
&\tau_2''(v_1, a, x) = \infty, \forall a : a > 1, \forall x \in \mathbb{R}^+ \\
&\tau_2''(v_i, q^j, r^k) = \min_{\substack{\alpha_1,\ldots,\alpha_d \\ \sum \alpha_j = 1}} \max \begin{cases} \tau_2''(p_1, q^{\lfloor j + \log_q \alpha_1 \rfloor}, L - l_1) + l_1' \\ \vdots \\ \tau_2''(p_d, q^{\lfloor j + \log_q \alpha_d \rfloor}, L - l_d) + l_d' \end{cases}
\end{aligned}
$$

We will state the two theorems about accuracy and runtime without proofs, since these are similar to the proofs of theorems 1 and 2.

Theorem 5. *Given L, let k be such that $\tau_2''(v_n, q^k, L_1) \leq L_2 < \tau_2''(v_n, q^{k+1}, L_1)$ and a be such that $\tau_2(v_n, a, L_1) \leq L_2 < \tau_2(v_n, a+1, L_1)$. Then $(1+\varepsilon)^{-1} \leq \frac{a}{q^k} \leq 1+\varepsilon$.*

Theorem 6. *Given maximum allowed path lengths L_1 and L_2 in two instances, if all edges have integer lengths, we can find k which satisfies $\tau_2''(v_n, q^k, L_1) \leq L_2 < \tau_2''(v_n, q^{k+1}, L_1)$ in time $O(mn^2\varepsilon^{-1} L_1 \log n)$, where m denotes the number of edges in the graph.*

Since the running time is linear in L_1, the algorithm that corresponds to the evaluation of the function τ_2'' is pseudo-polynomial.

2.4 FPTAS for counting small sums in $X_1 + X_2 + \ldots + X_n$

Turning the FPTASes for paths on directed acyclic graphs into an algorithm for counting sums in $X_1 + X_2 + \ldots + X_n$ works is analogous to the NP-hardness reduction from section 1. We transform the set of sets X_i into a directed acyclic graph with $n + 1$ vertices and $m := \sum_i |X_i|$ edges. We can therefore calculate the ε-approximation to the number of ρ-optimal solutions in time $O(mn^2\varepsilon^{-1} \log n)$ for a single instance, and in time $O(mn^3\varepsilon^{-1}\delta^{-1} \log n \log L_1)$ for two instances, if we allow multiplicative error of $(1 + \delta)$ for ρ.

3 Conclusion

References

1. Joachim M. Buhmann, Matus Mihalak, Rastislav Sramek, and Peter Widmayer. Robust optimization in the presence of uncertainty. In *Proceedings of the 4th conference on Innovations in Theoretical Computer Science*, ITCS '13, pages 505–514, New York, NY, USA, 2013. ACM.
2. D.B. Johnson and T. Mizoguchi. Selecting the k th element in x+y and x_1+x_2+âŃf+x_m. *SIAM Journal on Computing*, 7(2):147–153, 1978.
3. Ting Chen, Ming-Yang Kao, Matthew Tepel, John Rush, and George M Church. A dynamic programming approach to de novo peptide sequencing via tandem mass spectrometry. *Journal of Computational Biology*, 8(3):325–337, 2001.
4. Bingwen Lu and Ting Chen. A suboptimal algorithm for de novo peptide sequencing via tandem mass spectrometry. *Journal of Computational Biology*, 10(1):1–12, 2003.
5. Dalit Naor and Douglas Brutlag. On suboptimal alignments of biological sequences. In *Combinatorial Pattern Matching*, pages 179–196. Springer, 1993.
6. Daniel Štefankovic, Santosh Vempala, and Eric Vigoda. A deterministic polynomial-time approximation scheme for counting knapsack solutions. *SIAM Journal on Computing*, 41(2):356–366, 2012.