Università di Pisa
Dipartimento di Informatica

Technical Report: TR-01-04

An approximation algorithm for computing longest paths

Maria Grazia Scutellà

January 24, 2001

ADDRESS: Corso Italia 40, 56125 Pisa, Italy. TEL: +39 050 887111 FAX: +39 050 887226

An approximation algorithm for computing longest paths

Maria Grazia Scutellà *

January 24, 2001

Abstract

We show how the color-coding method by Alon, Yuster and Zwick [2], in its version specialized to compute simple paths of a specified cardinality k, can be extended to address the presence of arc lengths (the existence of such an extension was outlined by the authors for a more general color-coding algorithm, but it was not explicitely described, and its time complexity was not discussed).

The extension is then used to derive approximation results for the general longest path problem.

Keywords: Heuristics, longest path, color-coding method

1 Introduction

A randomized method, called the *color-coding* method, has been recently proposed by Alon, Yuster and Zwick [2] to find simple paths of a specified cardinality k (as well as other kinds of "small" subgraphs) in a given (undirected or directed) graph, where by cardinality of a path we mean the cardinality of its arc set. In particular the authors proved that, given a directed graph $G = (V, E)$, for any fixed k it is possible to determine in $O(2^k m \log n)$ time whether G contains a simple directed path of cardinality exactly k, where m is the cardinality of E and n is the cardinality of V.

One consequence is that it is possible to decide in polynomial time whether a directed graph G contains a simple directed path formed by $O(\log n)$ arcs.

Another consequence is that the color-coding method actually provides the best polynomially computable solution for the longest path problem in the case of equal arc lengths, i.e., when a simple directed path formed by the maximum number of arcs has to be determined. As known from the literature [3], both the general longest path problem and the special case of equal arc lengths are NP-hard; moreover, the longest path is not approximable within a constant approximation ratio [4]. In the special case, the color-coding method allows one to approximate the longest path with an approximation ratio which is bounded from below by $(\frac{logn}{n})$, which is currently the best approximation result for the longest path computation.

In their work, after presenting the color-coding method to compute simple (directed) paths, Alon, Yuster and Zwick generalize the method to find subgraphs characterized by a so-called bounded tree-width, a notion introduced by Robertson and Seymour [6]. They prove that, given a (directed or undirected) graph G, and given a graph H with k nodes and with a tree-width t, a subgraph of G isomorphic to H, if one exists, can be found in $O(2^k n^{t+1} logn)$ time. Since simple paths are special subgraphs with a bounded tree-width, having a tree-width equal to 1, the last result, when specialized to simple paths, provides a result which is sligthly worse than the previous one, allowing one to discover a simple path of cardinality k in $O(2^k n^2 logn)$ time.

After presenting their more general result, the authors observe that their algorithm could be adapted to compute the copy of H in G with the maximal/minimal total cost, in the case in which real costs are associated with the arcs of G. However, such an adaptation is not explicitely described, and its time complexity is not discussed.

In this paper we will explicitely show how the color-coding method, in its version specialized to compute simple paths of a specified cardinality k, can be extended, in a straightforward way, in order to address the presence of arc lengths in G. The algorithm extension will determine, at the same $O(2^k m \log n)$ time complexity of its decisional version, the longest path, from a given source node s to a given destination node t in a directed graph G, by restricting the attention to the simple directed paths of G of cardinality at most k, for any fixed k.

*Dipartimento di Informatica, Corso Italia 40, I-56125 Pisa (ITALY), e-mail: scut@di.unipi.it

An implication is that, given a directed graph G, the problem of computing a longest path in G from a source s to a destination t, among the simple directed paths of G whose cardinality is bounded from above by $\log n$, can be solved in $O(nm\log n)$ time.

The above result will be used to derive an approximation algorithm for the general longest path problem. Given a directed graph G, an algorithm will be proposed which approximates the longest path with an approximation ratio which is bounded from below by $(\frac{logn}{n\delta+logn})$, where δ denotes the maximum mean length among all the directed cycles of G, i.e., it is the maximum ratio between cycle length and cycle cardinality [1]. The obtained approximation ratio is parametric with respect to the maximum mean cycle length of G, i.e., a relationshiph is established between longest path value and maximum mean cycle length. In the case of graphs characterized by a constant value of δ, an $\Omega(\frac{logn}{n+logn})$ approximation ratio is thus derived. This result generalizes the best known result for the longest path problem in the case of equal arc lengths, which satisfies the above property; in fact, in that case it is $\delta = 1$.

If, in addition, the length of the simple directed paths from the source s to the destination t is an increasing function with respect to the cardinality of the path, as for the longest path problem in the case of equal arc lengths, the above approximation result can be improved to $\Omega(\frac{logn}{n})$; i.e., the best known approximation result for the corresponding decisional problem can be matched. At our knowledge, no approximation result has been derived before for the general longest path computation.

In the following, firstly the color-coding method will be reviewed, and an extension addressing the presence of arc lengths will be described (Section 2). Then, an approximation algorithm will be proposed for the general longest path computation (Section 3).

2 The color-coding method and its extension

2.1 The color-coding method for computing simple paths

Let $G = (V, E)$ be a directed graph, with $|V| = n$ and $|E| = m$. Given a source node s and a destination node t, and given a positive integer k, $k \leq n$, consider the problem of finding a simple directed path from s to t in G formed by exactly $k-1$ arcs. The idea of Alon, Yuster and Zwick [2] is to choose a random coloring of the nodes of G with exactly k colors, and determine whether a directed *colorful* path of cardinality $k-1$ exists in G which goes from s to t, where a path P is said to be colorful if the nodes of P are colored by distinct colors under the chosen random coloring of V. Clearly, each colorful path is a simple path, whereas the viceversa is not necessarily true.

Given a random coloring, say c, of the nodes of G with exactly k colors, the authors describe an algorithm which computes a colorful path from s to t in G of cardinality $k-1$ with respect to c, if one exists, in $O(2^k m)$ worst case time.

Since each simple path of cardinality $k-1$ has a probability $\frac{k!}{k^k} > e^{-k}$ to become colorful, the approach is in fact a randomized algorithm, discovering a simple directed path of cardinality $k-1$, if one exists, in $O(2^k m)$ expected time.

As outlined by the authors, the randomized algorithm can be derandomized thanks to the use of a k-perfect family of hash functions from $\{1, 2, \ldots, n\}$ to $\{1, 2, \ldots, k\}$, i.e., a list of colorings of V such that, for each subset V' of V such that $|V'| = k$ (and so, for each simple path from s to t of cardinality $k-1$), there is a coloring in the list which assigns a distinct color to each vertex in V'. Since there exists a family of hash functions of this type of size $O(2^k \log n)$, a deterministic algorithm can be derived which finds a simple directed path of cardinality $k-1$, if one exists, in $O(2^k m \log n)$ time. For details about k-perfect families of hash functions and about derandomization we refer to [2] and to [7].

The building block of the color-coding method is the algorithm which computes a colorful path from s to t in G of cardinality $k-1$, if one exists, with respect to a given coloring c from $\{1, 2, \ldots, n\}$ to $\{1, 2, \ldots, k\}$. Let us describe this algorithm.

Given a coloring c, consider the following dynamic programming approach. Starting from s, the algorithm determines, for each node $i \in V$, the set of colors on the colorful paths from s to i of cardinality h, $h = 1, 2, \ldots, k-1$. Assume that the collection, say $C^h(i)$, of the color sets on the colorful paths of cardinality h from s to i has been determined, for each node i: since only the different color sets are maintained for each node i (instead of the set of all the corresponding colorful paths), the cardinality of $C^h(i)$ is

$$O\begin{pmatrix} k \\ h \end{pmatrix}.$$

In order to compute $C^{h+1}(i)$ for each node i, i.e., the collection of the color sets on the colorful paths from s to i of cardinality $h+1$, the algorithm works as follows. For each node i, it considers each set $C \in C^h(i)$, and each arc (i,j) in the forward star of i: if $c(j) \notin C$, then $C \cup c(j)$ is added to $C^{h+1}(j)$, since it corresponds to a colorful path from s to j of cardinality $h+1$.

At termination, i.e., when the collection $C^{k-1}(i)$ has been computed for each node i, the input graph G contains a colorful path of cardinality $k-1$ from s to t if and only if $C^{k-1}(t)$ is not empty.

The dynamic programming approach performs at most $\sum_{h=1}^{k-1} h \binom{k}{h} m$ operations, which is $O(2^k m)$.

Observe that the algorithm can determine, at the same $O(2^k m)$ time complexity, whether a colorful path of cardinality h exists from s to t, for each cardinality $h \leq k-1$.

2.2 An extension

Assume now that a length $l(i,j)$ is associated with each arc (i,j) of G, and that, for some specified value k, a longest path has to be determined among all the simple directed paths from s to t which are formed by $k-1$ arcs, if paths of this kind exist.

Given a random coloring c from $\{1,2,\ldots,n\}$ to $\{1,2,\ldots,k\}$, consider the dynamic programming approach by Alon, Yuster and Zwick. Extend the approach as follows: for each color set $C \in C^h(i)$, for $i \in V$ and $1 \leq h \leq k-1$, introduce a label $l(C)$, which is the maximum length of the colorful paths from s to i which are formed by h arcs and that are colored by the colors in C. The observation is that, in order to compute a longest simple path of cardinality h, for $1 \leq h \leq k-1$, it is sufficient to maintain a single label for each color set $C \in C^h(i)$, for each node i.

As before, assume that the collection $C^h(i)$ of the color sets on the colorful paths of cardinality h from s to i has been determined, for each node i; moreover, assume that the corresponding labels have been computed: as in the decisional version, the cardinality of this collection is $O\binom{k}{h}$.

Describe now the typical algorithm iteration. For each node i, consider each set $C \in C^h(i)$ and its label $l(C)$: for each arc (i,j) in the forward star of i, if $c(j) \notin C$ and if $C \cup c(j)$ does not belong to $C^{h+1}(j)$, then add $C \cup c(j)$ to $C^{h+1}(j)$, with label equal to $l(C) + l(i,j)$; otherwise, if $c(j) \notin C$ and $C \cup c(j)$ already belongs to $C^{h+1}(j)$, then update its label if $l(C) + l(i,j) > l(C \cup c(j))$: $l(C \cup c(j)) := l(C) + l(i,j)$.

At termination, i.e., when the collection $C^{k-1}(i)$ has been computed for each node i, then the length of the longest colorful path of cardinality $k-1$ from s to t, with respect to the given random coloring c, is provided by the maximum label associated with the color sets in $C^{k-1}(t)$, if $C^{k-1}(t)$ is not empty.

In order to compute the maximum length among all the simple paths from s to t of cardinality $k-1$, it is sufficient to return the maximum among the labels associated with node t in $C^{k-1}(t)$, for each coloring c in the k-perfect family of hash functions used during the derandomization phase.

The algorithm correctness follows immediately.

Observe that, in order to find the colorful paths themselves, and therefore the longest simple path, we can maintain a predecessor, say $pred(C) = \langle j, C' \rangle$, for each color set $C \in C^h(i)$, $i \in V$ and $1 \leq h \leq k-1$, denoting the node j and the color set C' in $C^{h-1}(j)$ which provided the current updating of the label associated with C.

Therefore:

Theorem 2.1 *For any fixed k, the longest simple path of cardinality k from a source node s to a destination node t can be determined in $O(2^k m \log n)$ time.*

Corollary 2.1 *The problem of computing a longest path in a directed graph G, from a given source s to a given destination t, among the simple directed paths of G whose cardinality is bounded from above by $\log n$, can be solved in $O(nm \log n)$ time.*

Proof: It follows from the observation that, given a value k, the extension of the color-coding method allows one to compute, at the same time complexity, the maximum length of the simple paths of cardinality h, for each h such that $1 \leq h \leq k-1$. □

Clearly, similar results hold true in the case in which shortest simple paths have to be determined.

3 An approximation algorithm for the general longest path problem

Let $G = (V, E)$ be a directed graph with n nodes and m arcs. Let $l(i, j)$ be the integer positive length associated with the arc (i, j) of G. Given a source node s and a destination node t, the *longest path problem* consists of computing a longest simple path from s to t w.r.t. the arc lengths $l(i, j)$. This problem is NP-hard [3]; moreover, it is not approximable within a constant approximation ratio [4].

Hereafter we will propose a simple algorithm to approximate the longest path from s to t, which is based on the result described in Section 2.

The algorithm works as follows.

At the beginning, it computes the maximum mean length among all the directed cycles of G, i.e.,

$\delta = \max_C \frac{l(C)}{|C|}$,

where $l(C)$ denotes the length of the directed cycle C. Such a computation can be performed in $O(mn)$ time [5].

If $\delta \leq 0$, then the longest path problem can be solved to optimality by a standard shortest path algorithm on G; to this end, it is sufficient to compute a shortest path from s to t with respect to the costs $-l(i, j)$, for each arc (i, j) of G, due to the absence of negative cycles after performing the cost transformation. The longest path can thus be determined in $O(mn)$ time [1].

If $\delta > 0$, then subtract the value δ from the length of each arc of G; in other words, a length transformation is performed such that $l'(i, j) = l(i, j) - \delta$, for each arc (i, j) of G. Under the transformed length l', no positive directed cycle is present in G; then compute a longest path from s to t, with respect to the transformed length l', by applying a standard shortest path approach as described before for the case $\delta \leq 0$. Let P be the computed (simple) path from s to t, and k be its cardinality.

Two cases are addressed, depending on the cardinality k of P.

If $k \geq \lceil \log n \rceil$, then return P. Otherwise, use the extension of the color-coding method described in Section 2 to compute the longest path, say P', among the simple directed paths from s to t whose cardinality is bounded from above by $\lceil logn \rceil$, and return P'.

The following properties hold true.

Property 3.1 *The cardinality of the longest (simple) path from s to t is greater than or equal to k.*

Proof: Assume that the longest path, say P^*, is formed by h arcs. The length of P^* with respect to the transformed length l' is given by $l'(P^*) = l(P^*) - h\delta$, where $l(P^*)$ denotes the length of P^* under the original arc lengths. Since P is the longest path with respect to l', it is $l'(P) = l(P) - k\delta \geq l'(P^*) = l(P^*) - h\delta$. Being $l(P) \leq l(P^*)$, that implies $h \geq k$. □

Theorem 3.1 *The absolute error corresponding to the simple path P is bounded from above by $(n-k)\delta$.*

Proof: Let P^* denote the longest path, and assume that it is formed by h arcs, with $h \geq k$ (Property 3.1). Since $l'(P) = l(P) - k\delta \geq l'(P^*) = l(P^*) - h\delta$, it is $l(P^*) - l(P) \leq (h-k)\delta$, and therefore the absolute error is bounded from above by $(n-k)\delta$. □

Based on the above properties, let us estimate now the approximation ratio $w = \frac{z_A}{z_{Opt}}$, where z_A denotes the length of the path returned by the approximation algorithm, whereas z_{Opt} denotes the longest path value.

Theorem 3.2 *The algorithm guarantees an approximation ratio $\geq (\frac{logn}{n\delta + logn})$, in $O(nmlogn)$ time.*

Proof: Consider first the case $k \geq \lceil \log n \rceil$ (when this happens, path P is returned by the approximation algorithm). Due to Theorem 3.1, it is $z_{Opt} \leq z_A + (n - logn)\delta$, and so $w = \frac{z_A}{z_{Opt}} \geq \frac{z_A}{z_A + (n-logn)\delta}$. Observe that $\frac{z_A}{z_A+(n-logn)\delta}$ is an increasing function with respect to the argument z_A: therefore, since $z_A \geq logn$ (since $l(i,j) \geq 1$ for each $(i,j) \in E$), it gets its minimum for $z_A = logn$. We conclude that $w \geq \frac{logn}{(n-logn)\delta + logn}$.

Consider now the case $k < \lceil logn \rceil$ (when this happens, path P' is returned by the approximation algorithm). Also in this case, thanks to the extension of the color-coding method, which computes the longest path among all the simple paths with a number of arcs bounded from above by $\lceil logn \rceil$, it is $z_A \geq logn$.

Due to Theorem 3.1, it is $z_{Opt} \leq l(P) + (n-k)\delta$, where $l(P)$ is the length of the path P with respect to the original arc lengths. Since $l(P) \leq z_A$ and $z_{Opt} \leq l(P) + (n-k)\delta$, it is $w = \frac{z_A}{z_{Opt}} \geq \frac{z_A}{z_A + (n-k)\delta}$. By the previous argument we get the minimum for $z_A = logn$, and therefore $w \geq \frac{logn}{(n-k)\delta + logn}$. Since $k \geq 1$, we derive $w \geq \frac{logn}{(n-1)\delta + logn}$.

By combining both cases, it is $w \geq \frac{logn}{n\delta + logn}$.

The algorithm time complexity is due to the extension of the color-coding method, having an O$(nm\log n)$ time complexity (Corollary 2.1), whereas both the computation of value δ and the shortest path computation can be performed in O(mn) time. □

If δ is a constant or, in other words, the longest cycle of G has an O(n) length, then the approximation ratio is $\Omega(\frac{logn}{n+logn})$. This property is satisfied in the case of equal arc lengths, where it is $\delta = 1$ (after scaling the arc lengths to 1).

If, in addition, the length of the simple directed paths from the source s to destination t is a monotone function with respect to the path cardinality, in the sense that paths formed by more arcs have a higher length with respect to paths composed by a less number of arcs (as happens in the case of the longest path problem with equal arc lengths), then the estimate of the approximation ratio can be enhanced, by maintaing the same O$(nm\log n)$ time complexity.

Consider the approximation algorithm previously described. Due to the monotone property, if $k < \lceil logn \rceil$ then the extension of the color-coding method will return the longest (simple) path of cardinality $\lceil logn \rceil$, whose length is greater than $l(P)$ by at least $(logn - k)$. Therefore, it is $z_A \geq l(P) + (logn - k)$. The same result holding for the longest path problem in the case of equal arc lengths can then be derived.

Theorem 3.3 *In the case of a monotone cost function (when δ is a constant), the longest path is approximable by an approximation ratio which is $\Omega(\frac{logn}{n})$.*

Proof: Consider the proof of Theorem 3.2. In the first case ($k \geq \lceil logn \rceil$), it is $w \geq \frac{logn}{(n-logn)\delta + logn}$, as in the general case; being δ a constant, w is $\Omega(\frac{logn}{n})$.

Cosider now the case $k < \lceil logn \rceil$. By Theorem 3.1, it is $z_{Opt} \leq l(P) + (n-k)\delta$; moreover, due to the monotone property, it is $z_A \geq l(P) + (logn - k)$ or, equivalently, $l(P) \leq z_A - (logn - k)$. Therefore it is $w = \frac{z_A}{z_{Opt}} \geq \frac{z_A}{z_A - (logn-k) + (n-k)\delta}$. As before we get the minimum for $z_A = logn$, and thus $w \geq \frac{logn}{(n-k)\delta - (logn-k) + logn}$.

Being δ a constant value, also in this case w is $\Omega(\frac{logn}{n})$. □

References [1] R.K.Ahuja, T.L. Magnanti and J.B. Orlin, Network Flows: theory, algorithms and applications, Prentice Hall, New Jersey, 1993.

[2] N. Alon, R. Yuster and U. Zwick, Color-Coding, Journal of the ACM 42 (4) (1995) 844-856.

[3] M.R. Garey and D.S. Johnson, Strong NP-completeness results: motivation, examples and implications, J. Ass. Comp. Mach. 25 (1978) 499-508.

[4] D. Karger, R. Motwani and G.D.S. Ramkumar, On approximating the longest path in a graph, Algorithmica 18 (1997) 82-98.

[5] R.M. Karp, A characterization of the minimum cycle mean in a diagraph, Discrete Mathematics 23 (1978) 309-311.

[6] N. Robertson and P. Seymour, Graph minors. II. Algorithmic aspects of tree-width, J. Algorithms 7 (1986) 309-322.

[7] J.P. Schmidt and A. Siegel, The spatial complexity of oblivious k-probe hash functions, Siam Journal on Computing 19 (5) (1990) 775-786.