Shortest $H$-restricted paths in arc colored digraphs.

Barba Flores Luis Felipe$^1$  
Galeana-Sánchez Hortensia$^2$

November 16, 2010

Abstract

Consider a digraph $H$ with $k$ vertices possibly with loops and let $D$ be another digraph whose arcs are colored with the vertex set of $H$ (we call $H$ a guide digraph of $D$). An $H$-restricted path $T$ in $D$ will be a directed path in which, given two consecutive arcs in $T$, their colors, in the same order they appear on $T$, are an arc that belong to $A(H)$ i.e. if $T = (u_0, u_1, \ldots, u_n)$, then $(\text{color}(u_i), \text{color}(u_{i+1})) \in A(H)$ for all $i \in \{0, \ldots, n-1\}$.

We present an algorithm that solves the problem of finding a shortest $H$-restricted path between a fixed vertex $v$ and every other vertex in the digraph in $O(q \log p)$ time using $O(p^2)$ space and supposing $k$ constant.

Keywords: Shortest path algorithm, arc colored digraphs, $H$-restricted paths.

1 Introduction

Consider a digraph $H$ with $k$ vertices possibly with loops and let $D$ be another digraph whose arcs are colored with the vertex of $H$ (we call $H$ a guide digraph of $D$). An $H$-restricted path $T$ in $D$ will be a directed path in which, given two consecutive arcs in $T$, their colors, in the same order they appear on $T$, are an arc that belong to $A(H)$.

This construction is a generalization of the concept of alternating paths, a concept in which a path is only asked to have different colors between every two consecutive arcs. Properties of alternating paths have been well studied in different papers such as [2] [1], in the last one they present an algorithm that finds the shortest alternating path between two vertices in $O(n^2)$. A particular case with many applications, linked with bipartite graphs, is finding the shortest alternating bi-chromatic path between two vertices.

The concept of $H$-restricted path was first introduced by Linek and Sands [4] an then used by Arpin and Linek [3] in their work on $H$-independent and $H$-absorbent sets. Another result regarding this paths was founded by Galeana and Delgado [5] by introducing the concept of $H$-restricted kernel, extending the usual definition of kernel with $H$-restricted paths. However non algorithmic approach has been done in this regard.

Dijkstra presented an algorithm to find a shortest path between two vertices in a graph in $O(q + p)$ time using Breath First Search, and several other approaches has been done in this field based on the ideas in Dijkstra’s algorithm [6] [7]. We present a generalization of his algorithm that can be applied in many particular tasks such as:

- Finding shortest alternating paths in a digraph by using a complete guide digraph $H$ without loops.
- Finding shortest monochromatic paths by using isolated vertices in $H$.

---

$^1$ Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas, Universidad Nacional Autónoma de México  
$^2$ Instituto de Matemáticas, Universidad Nacional Autónoma de México
• Finding monotone increasing or decreasing directed paths by using a partial order as a guide digraph.

The last one referring to monotone paths have been a constant subject in the study of networks [8]. However, these are just some particular applications that provide us an idea of the versatility H-restricted paths have.

They also have many applications on scheduling problems, in which we have different states and, to go from one to another, specific tasks are required, but many tasks usually cannot be done unless another one has already been done, therefore an order between them can be imprinted in the guide digraph $H$ so that we can find a shortest sequence of tasks that lead us to our goal.

In this paper we introduce an algorithm to find a shortest $H$-restricted path between a fixed vertex $v$ and every other vertex of $D$ in $O(q \log p)$, where $q$ and $p$ are the number of arcs and vertices respectively and uses $O(p^2)$ space. We must suppose $k$ constant to get this complexity.

2 General concepts and notation

Given a digraph $D$ with $p$ vertices and $q$ arcs, denoted by $(p, q)$ digraph, then, if $e = (u, v) \in A(D)$ we say that $u$ incides toward $v$ and that the arc $e$ is incident with both vertices $u$ and $v$.

Let $C(D)$ be a $k$-coloration by arcs or just $k$-coloration of $D$ if $C(D) = (c_1, c_2, \ldots, c_k)$ is a partition of $A(D)$. This means that for any two elements $c_i, c_j \in C(D)$

$$c_i \cap c_j = \phi \text{ and } \bigcup_{i=0}^{k} c_i = A(D).$$

We can see that every arc of $D$ belongs to a unique element of $C(D)$, meaning we can define a function $c : A(D) \rightarrow C(D)$ so that, $c(e) = c_i$ if and only if $e \in e_i$, in this case we would say that $c(e)$ “is the color of $e$”. Given a vertex $v \in V(D)$ the color $c_i$ is incident with $v$ if and only if exists $e \in A(D)$ so that $e$ is incident with $v$ and $c(e) = c_i$.

Let $D$ be a digraph possibly with loops with a $k$-coloration by arcs, we will call a digraph $H$, guide digraph of $D$, if and only if $V(H) = C(D)$, we must note that $H$ should not be a multigraph since it would not add any new restrictions to the $H$-restricted paths.

Now if $T = (v_0, v_1, \ldots, v_n)$ a directed path in $D$ we will say $T$ is $H$-restricted if and only if for every $i \in \{0, 1, \ldots, n-2\}$ it stands that:

$$(c((v_i, v_{i+1})), c((v_{i+1}, v_{i+2}))) \in A(H). \quad (1)$$

In which case we will say that the arcs $(v_i, v_{i+1})$ and $(v_{i+1}, v_{i+2})$ are compatibles.

With this we can now define, given two vertex $u, v \in V(D)$, the $H$-distance or H-distance $\delta_H(u, v)$ as the length of a minimum $H$-restricted $uv$-path in $D$ if it exists, if not $\delta_H(u, v) = \infty$.

3 The Algorithm

The following algorithm will let us decide if it exists an $H$-restricted path in a digraph $D$ colored by arcs with $H$ as a guide digraph, between a fixed vertex $u$ and any other vertex $v$ in the digraph $D$, and in case it exists the algorithm will present it explicitly.

Given a directed $uv$-path $T$ in $D$ we will use the notation $\text{end}(T)$ to refer us to $w$, the end of $T$. The last arc of $T$, i.e. the one that incides toward $w$ in $T$, will be called terminal segment or last segment. Also we will define the function $PARENT : V(D) \times \{T \mid T \text{ directed path of } D\} \rightarrow V(D) \cup \{\phi\}$, such that $PARENT(x, T)$ represents the vertex which incides toward $x$ in the directed path $T$ in case it exists, if not the function returns $\phi$. 

\hspace{1cm}2
The main idea of this algorithm is to extend the Breath First Search algorithm (BFS), such that it follows a color pattern defined in the guide digraph $H$.

In a normal BFS we pass once through each vertex, in this algorithm the difference is mainly that we will analyze each vertex at most once for each color that is incident with it. Other difference we must acknowledge is that the queue $Q$, used in BFS, in this algorithm will contain full directed paths instead of just vertices, such that it will allow us, at the moment of analyze its end, to know from which color we have arrived, furthermore it will give us explicit $H$-restricted paths at the end of the algorithm.

For the operation of adding a path $T$ to the end of the queue $Q$ we will use the following notation $Q ← Q + T$.

Figure 1:

In this figure we must notice that if the vertices of $D$ are visited, taking 1 as the root and using BFS, then we will reach 2 by the arc $(1, 2)$, nevertheless the directed path $T : (1, 2, 6)$ is not $H$-restricted, from which stands that if the algorithm restricts us to pass only once through each vertex, then we will never reach the vertex 6, it is from here that we must design the new algorithm such that it passes more than once through each vertex, in fact, at most once for each color than is incident with it. By doing this, at some point in the performance of the algorithm we will be able to construct the $H$-restricted path $P : (1, 3, 4, 2, 6)$.

Algorithm 1 Minimum $H$-distance algorithm

[Given a $(p, q)$ digraph $D$ colored by arcs with a guide digraph $H$. Algorithm to find $\delta_H(u, v)$ for a fixed vertex $u$ and every other vertex $v$.]

0. [This step is performed only once in the algorithm. Inside it we define the array $PATH$, that will represent for each vertex $v$ of $D$, the minimum $H$-restricted $uv$-path found by the algorithm. The array $COLORS$ which will represent the set of colors that are incident with $v$. And the array $\lambda$ which will represent the $H$ distance between $u$ and $v$. Finally we will define the queue $Q$ as the one who initially consists of the trivial directed $uu$-path.] Let $PATH$, $COLORS$ and $\lambda$ be three arrays of size $p$, and let $Q$ be a queue.

$\forall v \in V(D)$ if $v \neq u$, then $\lambda(v) ← \infty$, $PATH(v) ← \phi$ and $COLORS(v) ← \{c(e) | e is incident with v\}$.

$\lambda(u) ← 0$, $PATH(u) ← T : u$.

$Q ← \{T : u\}$

1. [In this step if $Q$ is not empty we remove a directed path from the beginning of the queue and send it to step 2. If $Q$ is empty that means that the algorithm is done, in which case it displays the results.] If $Q \neq \phi$, then we remove from the beginning of $Q$ a directed path $T$.

If $Q = \phi$ terminate the algorithm and display the results.
2. [In this step we analyze the path $T$ obtained from step 1. Let $x$ be the end of $T$, for every arc $e = (x, v)$, if the $H$-restricted path $T$ can be extended with $e$ to a new $H$-restricted path $T'$, and the end of it has not been reached in the algorithm with another path with the color of its last segment equal to $c(e)$, then we add $T'$, the constructed path to $Q$ and we update the array $\text{COLORS}$ for the end of $T'$, that is removing from the set $\text{COLORS}(\text{end}(T'))$ the color $c(e)$, so that the next time a path, with the color of its last segment equal to $c(e)$ is analyzed in this step it will not be added to $Q$.

If the constructed path improves (is shorter) the previously found for its end or there was not any, then we update the arrays $\lambda$ and PATH for the end of the new directed path $T'$.

Given $x = x_{\text{exit}}(T)$
\[ \forall e = (x, w) \in \text{A}(D) \text{ such that } w \notin V(T), \]
If $x = u$ or $c((\text{PARENT}(x, T), x)), c(e)) \in A(H)$, then:
\[ \text{if } c(e) \in \text{COLORS}(w), \text{then } \text{COLORS}(w) \leftarrow \text{COLORS}(w) - \{c(e)\}, \]
\[ T' \leftarrow T + e, Q \leftarrow Q + T', \text{ furthermore if } \text{long}(T') < \lambda(w), \text{then } \lambda(w) \leftarrow \text{long}(T') \text{ and PATH}(w) \leftarrow T'. \]

Return to Step 1.

For the next theorem we will denote $T_k$ as the $k$-th directed path removed from $Q$ in the Step 1 of the algorithm. Let us notice that $T_1$ will be the trivial $uu$-path initially contained by $Q$.

**Theorem 2** Given a digraph $D$ colored by arcs with a guide digraph $H$. After performing the algorithm 1, given $T_i$ it follows that for all $T_j$ if $i < j$, then $\text{long}(T_i) \leq \text{long}(T_j)$.

**Proof.**

We will make the proof by induction over $i$.

Base of the induction: If $i = 1$ then we know that $T_i$ is the trivial $uu$-path of length zero, therefore every other $T_j$, with $j > i$ is going to be such that $0 = \text{long}(T_i) \leq \text{long}(T_j)$ with which we have the base case.

Let $T_i$ be the $i$-th path removed from $Q$ and let us suppose that given $T_k$ with $k < i$ it follows that for all $T_j$ if $k < j$, then $\text{long}(T_k) \leq \text{long}(T_j)$. Now we prove the property for $i$, in order to do that we use another induction, now over $j$ with $i$ fixed.

If $j = i + 1$ (base of the induction), then we have two cases:

**Case 1:** $T_j$ was in $Q$ at the time $T_i$ was removed, therefore $T_j$ was constructed by adding an arc to the end of a path $T_k$, with $k < i$, in the same way $T_i$ was constructed by adding an arc to the end of another path $T_h$, and since $T_i$ was added to $Q$ before $T_j$, then $T_h$ was removed from $Q$ before $T_k$ or $h = k$, which implies that $h \leq k < i$.

With that facts we know by induction hypothesis that $\text{long}(T_h) \leq \text{long}(T_k)$ and this implies that $\text{long}(T_i) = \text{long}(T_h) + 1 \leq \text{long}(T_k) + 1 = \text{long}(T_j)$.

**Case 2:** $T_j$ was not in $Q$ at the time $T_i$ was removed, nevertheless $T_j$ is the next path being removed from $Q$, hence it had to be constructed from $T_i$, because between the removal of $T_i$ and the one of $T_{i+1} = T_j$ only paths created adding arcs to the end of $T_i$ where added to $Q$. Therefore $\text{length}(T_j) = \text{length}(T_i) + 1 > \text{length}(T_i)$.

Let us suppose the result for all $k$, such that $i + 1 \leq k < j$, and let us prove the result for $j$. Same as with the base of the induction we have two cases:

**Case 1:** $T_j$ was already in $Q$ at the time $T_i$ was removed, hence, the proof is equal to the case 1 of the base of the induction over $j$.

**Case 2:** $T_j$ was not in $Q$ at the time $T_i$ was removed, therefore $T_j$ was added to $Q$ after the removal of $T_i$. Which means that $T_j$ was constructed by adding an arc to the end of a $T_m$ with $i \leq m < j$. If $m = i$, then it is clear that $\text{length}(T_i) < \text{length}(T_i) + 1 = \text{length}(T_j)$. Otherwise, if $i < m$, by inductive hypothesis we know that $\text{length}(T_i) \leq \text{length}(T_m)$, moreover $\text{length}(T_j) = \text{length}(T_m) + 1$. 

4
Hence \( \text{length}(T_i) \leq \text{length}(T_m) \leq \text{length}(T_j) \).

\[ \Box \]

**Theorem 3** Given a digraph \( D \) colored by arcs with a guide digraph \( H \). After performing the algorithm 1, if \( T : (u = v_0, v_1, \ldots, v_k = v) \) is an \( H \)-restricted \( uv \)-path in \( D \), then we can assume that in the algorithm it was added to \( Q \) an \( H \)-restricted \( uv \)-path \( P \) in \( D \), such that \( \text{length}(P) \leq \text{length}(T) \) and they match in the color of their last segment.

**Proof.** By induction over \( \text{length}(T) \).

Let \( T \) be an \( H \)-restricted path in \( D \).

If \( \text{length}(T) = 1 \), then \( T = (u, v) \) and hence the first time that the Step 2 of the algorithm was performed, because \( v \) is adjacent from \( u \) and \( c((u, v)) \in \text{COLORS}(v) \), then \( T \) was added to \( Q \) and \( \text{path}(v) = T \), thereby \( \lambda(v) = \text{length}(T) = 1 \).

Let us suppose the result for all \( k < n \) and let \( T : (u = v_0, v_1, \ldots, v_{n-1}, v_n) \) be an \( H \)-restricted path such that \( \text{length}(T) = n \), we will see that the result stands for \( T \).

Let \( T' : (u = v_0, v_1, \ldots, v_{n-1}) \) be an \( H \)-restricted path constructed with the removal of the last segment of \( T \), now for inductive hypothesis we know that an \( H \)-restricted \( uv_{n-1} \)-path \( P' \) was added to \( Q \), with the color of its last segment matching the one of \( T' \) and \( \text{length}(P') \leq \text{length}(T') \). Thereby at the moment that \( P' \) is analyzed by the Step 2 of the algorithm, since \( v_{n-1} \) is adjacent from \( v_{n-1} \), then \( P' = P' \cup (v_{n-1}, v_n) \) is an \( H \)-restricted path, because \( T' \) was and they match in the color of their last segment; now we have two cases to review for that same moment in the algorithm:

**Case 1:** If \( c((v_{n-1}, v_n)) \in \text{COLORS}(v_n) \), then \( P \) was added to \( Q \), an \( H \)-restricted \( uv_n \)-path with the color of its last segment matching the one of \( T \). Moreover \( \text{length}(P) = \text{length}(P') + 1 \leq \text{length}(T') + 1 = \text{length}(T) \).

**Case 2:** If \( c((v_{n-1}, v_n)) \notin \text{COLORS}(v_n) \), it means that an \( H \)-restricted \( uv_n \)-path \( R \) has already been added to \( Q \), a path with the color of its last segment matching the one of \( T \) and \( P \). Because \( R \) was added to \( Q \), we know that \( R \) was constructed by adding an arc to another path \( R' \) in the Step 2 of the algorithm. Since \( R' \) was removed from \( Q \) before \( P' \), by 2 we have that \( \text{length}(R') \leq \text{length}(P') \), which implies that:

\[
\text{length}(R) = \text{length}(R') + 1 \leq \text{length}(P') + 1 \leq \text{length}(T') + 1 = \text{length}(T).
\]

Therefore in both cases we have an \( H \)-restricted \( uv_n \)-path shorter than \( T \), that match with \( T \) in the color of their last segment and such that it was added to \( Q \). Let us notice that when this happens, by the construction of the algorithm, it stands that \( \lambda(v_n) \leq \text{length}(P) \leq \text{length}(T) \). Thereby it stands the proof of this theorem.

\[ \Box \]

**Theorem 4** Given a digraph \( D \) colored by arcs with a guide digraph \( H \). After performing the algorithm 1 it follows that, for all vertex \( v \) in \( V(D) \), \( \lambda(v) = \delta_H(u, v) \). Moreover \( \text{PATH}(v) \) is a minimum directed \( H \)-restricted \( uv \)-path.

**Proof.** Given \( v \in V(D) \) we can acknowledge that at the end of the algorithm, because the arrays \( \text{PATH} \) and \( \lambda \) are modified at the same time, then \( \lambda(v) = \infty \) if and only if \( \text{PATH}(v) = \phi \), also we must remark that by definition \( \lambda(v) = \text{length}(\text{PATH}(v)) \).

Let us see that if there exists an \( H \)-restricted \( uv \)-path \( T \) in \( D \), then by the previous theorem we have that \( \text{PATH}(v) \neq \phi \) and this happens if and only if \( \lambda(v) \neq \infty \). Therefore we just proved using the opposed, that if \( \lambda(v) = \infty \), then it does not exist any \( H \)-restricted \( uv \)-path in \( D \).
Now if $T$ a minimum $H$-restricted $uv$-path, then, by theorem 3, an $H$-restricted path $P$ entered the set $Q$, such that $\text{length}(P) \leq \text{length}(T)$ and $\lambda(v) \leq \text{length}(P)$, hence as $T$ is minimum:

$$\text{length}(T) \leq \text{length}(\text{PATH}(v)) \leq \text{length}(P) \leq \text{length}(T) = \delta_H(u,v).$$ (3)

Also from here we can assume that:

$$\lambda(v) = \text{length}(\text{tray}(v)) = \text{length}(P) = \text{length}(T) = \delta_H(u,v)$$ (4)

Therefore $\text{PATH}(v)$ is also minimum and $\lambda(v) = \delta_H(u,v)$. □

Thanks to the previous theorem we know that the algorithm works the way we wanted, now it only rests to review its complexity, to do that we will analyze each step of the algorithm and find out how many times they are performed.

The step 0 is performed an unique time with a complexity of $O(p)$, since for every vertex of $D$ many assignments of variables take place. The complexity of the step 1 is $O(1)$, so the only thing we must look over is how many times it is performed, to do that let us notice that step 2 returns to step 1 after being performed, then this last one will be performed as many times as the step 2 does. Therefore only remains to review the step 2.

In the last step of the algorithm each time a path $T$ enters, all the vertex adjacent to its end and the $H$-restricted paths constructed with them are reviewed, checking that no cycle is formed or else we would not have a path, this is done by checking that vertex adjacent to the end already belongs to $T$, to do that the best way to proceed is using a binary search over the vertices of $T$, this is done in $O(\log p)$ time. The rest of the comparisons and assignments in this step run in constant time.

Also, by construction of the algorithm, for each vertex $v$ in $D$ other than $u$, at most we will have $|\{c(e) \mid e \text{ is incident with } v\}| \leq |V(H)|$ different $H$-restricted $uv$-paths entering the step 2, which tell us that, at most, this step will be performed $|V(H)|$ times for each vertex and each time a $uv$-path is analyzed we will check $\delta^+(v)$ vertices adjacent to its end.

It is from here that we can now count the total amount of vertices adjacent to the end of a path analyzed in this step, to do that we will sum them all:

$$\sum_{v \in V(D)} |V(H)| \delta^+(v) = |V(H)| \sum_{v \in V(D)} \delta^+(v) \leq q|V(H)|$$

This follows from the fact that, in each iteration of the step 2, we analyze all the vertices adjacent from the end element of a path and, we analyze a path with an end vertex $v$ at most $|V(H)|$ times. Hence the analyze of the adjacent vertex in step 2 is performed $O(q)$ times and each time it takes $O(\log p)$, giving us a complexity for the step 2 and therefore for the hole algorithm of $O(q \log p)$.

To verify the amount of space used we must only notice that for each vertex in $D$ at must $|V(H)|$ $H$-restricted paths will be stored in $Q$, therefore as every path needs $O(p)$ space then the hole algorithm uses $O(p^2)$ space.
Figure 2:
In this figure we can observe the running of the algorithm 1 in the digraph $D$ with guide digraph $H$. We define our initial vertex $u$ and we start the process with $Q$ containing only the trivial directed $uu$-path. In step 1 the vertex adjacent to the end element of the just removed from $Q$ $uu$-path is analyzed, obtaining in this way two new directed $H$-restricted paths which are added to $Q$. The process continues until step 5, where we find that we can not go from green to red using the arc $(v,x)$, therefore $x$ would be isolated if we restrict our algorithm to pass only once through each vertex, nevertheless in step 9 $v$ is analyzed again and we reach a vertex that in other way would remain isolated. The algorithm ends in step 10 when finding itself with an empty queue $Q$. 
4 \( H \)-restricted paths in networks.

A network \( R \) is defined as a digraph with weight in the arcs, that is a digraph to which, through a function of assignment \( w : A(R) \rightarrow \mathbb{N} \), we give numeric values to the arcs of \( R \), consequently, given \( e \in A(R) \) we refer to \( w(e) \) as the weight of \( e \).

The approaching to networks in this article is not obvious, but if we think our color palette as a subset of \( \mathbb{N} \) and with it we paint the arcs of \( R \), then we will be able to define a function of assignment over the arcs of \( R \), so that every arc is associated with a unique numeric value, to which we will refer to as its weight.

![Figure 3](image)

In this figure we can observe a digraph being colored by a palette thought as a subset of \( \mathbb{N} \). As well as its guide digraph \( H \) representing the order inherited by \( \mathbb{N} \).

Equally we may define a guide digraph as shown in the example 4, more over, if we define \( H \) such that it represents the transitive order inherited by \( \mathbb{N} \), i.e. that for every \( c_i, c_j \in C(R) \subset \mathbb{N} \) the arc \( (c_i, c_j) \in A(R) \) if and only if \( c_i < c_j \), then, at the moment of performing the algorithm 1, we will obtain strictly crescent \( H \)-restricted paths in \( R \), in the same way we can modify \( H \) by reversing the direction of its arcs, such that we will obtain strictly decrescent \( H \)-restricted paths. If we also accept loops (arcs between the same vertex) in \( H \) on every vertex, then after performing the algorithm we will get crescent or decrescent \( H \)-restricted paths in \( R \).

It is noteworthy that we can use not only subsets of \( \mathbb{N} \) to color the arcs of the digraph, but any subset of any partial order, it is from this fact that the name of “monotone paths” emerged in the field of study of networks.

References


(2008):


[839] de Teresa, L. and Zuazua, E. Identification of the class of initial data for the insensitizing control of the heat equation, 15 p.


(2009):


Galeana-Sánchez, H. and Olsen, M. *Kernels by monochromatic paths in digraphs with covering number 2*, 13 p.


Goldfeder, I. A. *(k, l)-kernels in quasi-transitive digraphs*, 5 p.

(2010):


Bokowski, J., Bracho J. and Strausz, R., *Carathéodory-type theorems à la Bárány*, 10 p.


Galeana-Sánchez, H. and Goldfeder, I.A. *Counterexamples to a conjecture by Bang-Jensen*, 4 p.


Benabdallah, A., Cristofol, M., Gaitan, P. and de Teresa, L. *Controllability to trajectories for some parabolic systems of three and two equations by one control force*, 25 p.


