Distribution Agreement

In presenting this thesis or dissertation as a partial fulfillment of the requirements for an advanced degree from Emory University, I hereby grant to Emory University and its agents the non-exclusive license to archive, make accessible, and display my thesis or dissertation in whole or in part in all forms of media, now or hereafter known, including display on the world wide web. I understand that I may select some access restrictions as part of the online submission of this thesis or dissertation. I retain all ownership rights to the copyright of the thesis or dissertation. I also retain the right to use in future works (such as articles or books) all or part of this thesis or dissertation.

Signature:

SIYUAN	Date

Analysis of Detour Gap Numbers

By

Siyuan Lou
Master of Science

Computer Science

Michelangelo Grigni
Advisor

James Lu
Committee Member

Li Xiong
Committee Member

Accepted:

Lisa A. Tedesco, Ph.D.
Dean of the James T. Laney School of Graduate Studies

Date

Analysis of Detour Gap Numbers

By

Siyuan Lou
B.S., Tsinghua University, 2010

Advisor: Michelangelo Grigni, Ph.D.

An abstract of
A thesis submitted to the Faculty of the
James T. Laney School of Graduate Studies of Emory University
in partial fulfillment of the requirements for the degree of
Master of Science
in Computer Science
2012

Abstract

Analysis of Detour Gap Numbers
By Siyuan Lou

Spanner graphs appear in approximation schemes for problems such as the Traveling Salesman Problem, where an edge weighted graph defines a metric on its vertex set. In such schemes, a spanner is a subgraph of the input graph, which still represents nearly the same metric. We bound its total edge weight using the "detour gap number", which is defined by a linear program. In this thesis, we simplify this linear program, and state the complementary slackness conditions relating it and its dual. We also give a way to prune a graph based on those properties and a counter example showing the detour gap number is not monotone under edge deletion. The thesis also introduced a software package to facilitate the calculation of detour gap numbers.

Analysis of Detour Gap Numbers

By

Siyuan Lou
B.S., Tsinghua University, 2010

Advisor: Michelangelo Grigni, Ph.D.

A thesis submitted to the Faculty of the
James T. Laney School of Graduate Studies of Emory University
in partial fulfillment of the requirements for the degree of
Master of Science
in Computer Science
2012

Contents

List of Figures

Chapter 1

Introduction

Suppose $G = (V, E)$ is a connected graph where each edge $e \in E$ has a weight $w(e) \geq 0$. The weight of each edge can also be interpreted as its length. We denote the length of the shortest path between two vertices $u, v \in V$ as $d_G(u, v)$. If $(u, v) = e \in E$, we can also write $d_G(u, v)$ as $d_G(e)$. It is easy to see that d_G is a metric for G: it is symmetric, and for any three vertices $u, v, w \in V$, we have $d_G(u, v) \leq d_G(u, w) + d_G(w, v)$.[1]

Given a metric on n vertices, the metric Traveling Salesman Problem is to find a cyclic ordering of vertices such that the total length of the tour is the smallest according to its metric. In our case, it is to find a shortest tour, according to d_G, visiting all vertices in V and returning back to its starting point. The exact metric TSP was proved to be NP-hard by Karp in 1972 [2]. From then on, much effort has been devoted in seeking a polynomial time approximate algorithm to solve the Traveling Salesman Problem. Gutin's book [1] covers many important research areas of study on TSP including variations of

[1] Technically d_G is a "pre-metric" since we allow $d_G(u, v) = 0$ when $u \neq v$.

approximation schemes. In particular, the articles [3, 4, 5, 6] find approximate schemes in planar graphs. A polynomial time approximation scheme (PTAS) can be described as: given a metric and $\epsilon > 0$, find a tour with cost at most $1 + \epsilon$ times optimal, in time polynomial in $|V|$. Such schemes were found for metrics defined by certain families of graphs, e.g. planar graphs [3, 4]. A key step in these schemes is to find a light subgraph G' of G where we can compute the approximate shortest tour in polynomial time. This leads to the notion of a *spanner* of G.

A spanning subgraph of G is $G' = (V, E')$, where $E' \subset E$ and G' inherits its edge weighting from G. Similarly, G' defines its own metric $d_{G'}$ on V. Obviously, $d_{G'}(u, v) \geq d_G(u, v)$ for all vertices pair $(u, v), u, v \in V$, since we remove some edges from G to get G'. For some $r \geq 1$, if $d_{G'}(u, v) \leq r \cdot d_G(u, v)$ for all $u, v \in V$, then G' is a *r-spanner* of G. In approximate schemes for TSP, we are interested in finding a $(1 + \epsilon)$-*spanner* of G where ϵ is small. Given $r \geq 1$, we

Algorithm 1 ([7]) $Span(G = (V, E), r)$

$G' = (V, E')$, where initially $E' \leftarrow \emptyset$

for all $e \in E$ in non-decreasing w order **do**

  if $r \cdot w(e) < d_{G'}(e)$ **then**

    add e to E'

  end if

end for

return G'

can compute an *r-spanner* G' in G by the greedy algorithm (Algorithm 1) of Althöfer *et al.* [7]. In [8], it was also mentioned that if $G' = Span(G, r)$, then $G' = Span(G', r)$ and G' contains a minimum spanning tree of G. Generally,

if some connected graph G satisfies $G = Span(G, r)$, or we say it is a *greedy r-spanner graph*. We are interested in putting an upper bound on the relative total edge weight of such graphs, see Theorem 1. The ratio of the total edge weight of G and that of its minimum spanning tree T appears in the exponent of the time complexity of a PTAS.

Given a graph $G = (V, E)$ with a real weighting w, suppose T is a minimum spanning tree of G. For each edge $e \notin T$, we define the *detour gap number* of an edge (sometimes we say *detour gap* or simply *gap* for short) as in [8]:

$$g(e) \stackrel{\text{def}}{=} d_{G-e}(e) - w(e), \tag{1.1}$$

where d_{G-e} is the distance metric for the graph G removing e. Since $e \notin T$, we know $d_{G-e}(e)$ is finite. The *detour gap* measures the cost of removing an existing edge from graph G. In [8], the authors give the following theorem (Theorem 1) to bound the total weight of such a greedy spanner graph by the sum of the detour gaps of all its non-tree edges. We denote the total gaps of all non-tree edges as

$$g(G - T) \stackrel{\text{def}}{=} \sum_{e \in G-T} g(e),$$

where $G - T$ is G removing all edges in T. If we rescale the edge weights in G so that $w(T) = 1$, the theorem states:

Theorem 1. ([8]) *If $G = Span(G, 1 + \epsilon)$ and $w(T(G)) = 1$, then*

$$w(G) \leq 1 + (1/\epsilon) \cdot g(G - T).$$

Now if we fix G, T and ϵ, we can see that $w(G)$ is bounded by $g(G - T)$. In [9] the *ϵ-detour gap number* of graph G is introduced as:

$$gap_\epsilon(G, T) \stackrel{\text{def}}{=} \max g(G - T).$$

where the maximum is taken over all weighting w of G such that $w(T) = 1$ and $G = Span(G, 1+\epsilon)$, and T is an MST. We can see that if $gap_\epsilon(G,T)$ exists, it is an upper bound for the total weight of a greedy spanner graph.

To calculate $gap_\epsilon(G,T)$, we first define some auxiliary variables or notations. For some $e \in G-T$, T_e denotes the path in T that connects the two end points of e. We say some tree edge s crosses e, or $s \triangleleft e$, if $s \in T_e$. In G, we say P_e is a *detour path* of e if P_e and e form a simple cycle. All such *detour pairs* form a set $\mathcal{P} = \{(e, P_e) | P_e \text{ is a detour path of } e, e \in G-T\}$. For some e, we put all its detour paths (if exist) in sequence and give each one a subscript as $P_{e,i},\ i = 1,2,3,....$ Then $gap_\epsilon(G,T)$ can be calculated by the following linear program [9]:

$$
\begin{aligned}
gap_\epsilon(G,T) = \max_{w,g}\ & g(G-T) \\
& w(e) \geq 0 && \forall\, e \in G && \text{(1.2a)} \\
& w(T) \leq 1 && && \text{(1.2b)} \\
& w(s) \leq w(e) && \forall\, e \in G-T,\ \forall s \triangleleft e && \text{(1.2c)} \\
& g(e) \leq w(P_{e,i}) - w(e) && \forall\, (P_{e,i}, e) \in \mathcal{P} && \text{(1.2d)} \\
& g(e) \geq \epsilon \cdot w(e) && \forall\, e \in G && \text{(1.2e)}
\end{aligned}
$$

.

By observing the constraints, it is easy to see that at some optimal point (w, g), we always have

$$w(T) = 1$$

and

$$g(e) = \min_i w(P_{e,i}) - w(e), \forall\, e \in G - T.$$

That is, at optimal points, we always have the total weight of the minimum spanning tree (MST) scaled to 1 and g is determined by w as in equation (1.1).

This linear program is rather difficult to work with, so we introduce two simplified upper bounds that we obtain by removing constraints. By setting $\epsilon = 0$ in the linear program, we get a simpler linear program:

$$\begin{aligned} gap_0(G,T) = \max_{w,g}\ & g(G-T) \\ & w(e) \geq 0 && \forall\, e \in G \\ & w(T) \leq 1 \\ & w(s) \leq w(e) && \forall\, e \in G-T,\ \forall s \lhd e \\ & g(e) \leq w(P_{e,i}) - w(e) && \forall\, (P_{e,i}, e) \in \mathcal{P} \\ & g(e) \geq 0 && \forall\, e \in G \end{aligned} \quad . \tag{1.3}$$

By comparing these two linear programs, we can see that

$$gap_\epsilon(G,T) \leq gap_0(G,T)$$

since the latter is less constrained. Therefore, $gap_0(G,T)$ would be an upper bound for $w(G)$ in Theorem 1, and it may be easier to compute.

By omitting the MST constraints, we get an even simpler linear program:

$$\begin{aligned} gap'_0(G,T) = \max_{w,g}\ & g(G-T) \\ & w(e) \geq 0 && \forall\, e \in G \\ & w(T) \leq 1 \\ & g(e) \leq w(P_{e,i}) - w(e) && \forall\, (P_{e,i}, e) \in \mathcal{P} \\ & g(e) \geq 0 && \forall\, e \in G \end{aligned} \tag{1.4}$$

Note that

$$gap_0(G,T) \leq gap'_0(G,T),$$

so $gap'_0(G,T)$ is also an upper bound for the total edge weight of a greedy spanner graph. Besides, $gap'_0(G,T)$ shows some capability in bounding the spanner weight of some graph family [9]. For example, $gap'_0(G,T)$ is at most 2 for a planar graph.

This thesis simplifies the linear programs (1.3) and (1.4), and then analyzes the corresponding dual problem of each, the charging scheme, showing the complementary slackness conditions that relate an optimal primal solution and an optimal dual solution. After that, it introduces ways to prune a graph to show its basic structure determining its *detour gap number*. Finally, the thesis also introduces a software package to facilitate the computation of such primal and dual solutions.

Chapter 2

The Primal Problem

2.1 Simplification

For (1.3) and (1.4), we want to prove that the constraints

$$g(e) \geq 0, \ \forall \ e \in G$$

are not necessary. In other words, we want to show that they are equivalent to the following two linear programs:

$$\begin{aligned} rgap_0(G,T) = \max_{w,g} \ & g(G-T) & \\ w(e) \geq 0 \quad & & \forall \ e \in G \\ w(T) \leq 1 \quad & & \\ w(s) \leq w(e) \quad & & \forall \ e \in G-T, \ \forall s \lhd e \\ g(e) \leq w(P_{e,i}) - w(e) \quad & & \forall \ (P_{e,i}, e) \in \mathcal{P} \end{aligned} \tag{2.1}$$

and

$$\begin{aligned} rgap'_0(G,T) = \max_{w,g}\ & g(G-T) \\ & w(e) \geq 0 \qquad\qquad \forall\, e \in G \\ & w(T) \leq 1 \\ & g(e) \leq w(P_{e,i}) - w(e) \quad \forall\, (P_{e,i}, e) \in \mathcal{P} \end{aligned} \tag{2.2}$$

First we prove the equivalence of (1.3) and (2.1); that is,

$$rgap_0(G,T) = gap_0(G,T).$$

A similar argument applies to the other two as well. In order to prove our claim, we first introduce the definition of criticality.

Definition 1. *In some weighting w of G, an edge $f \notin T$ is* critical *to another edge $e \notin T$ ($e \neq f$) if when we increase the weight of f, i.e. $w(f)$, keeping the weights of other edges unchanged, the gap of e, i.e. $g(e)$ or equivalently, $d_{G-e}(e)$ also increases.*

Apparently, any feasible solution for (1.3) is also feasible for (2.1) since (1.3) contains all constraints of (2.1). Thus,

$$gap_0(G,T) \leq rgap_0(G,T). \tag{2.3}$$

To show equivalence, we only need to prove that there exists at least one optimal solution (w, g) for (2.1) such that

$$\forall\, e \in G - T,\ g(e) \geq 0.$$

Note that according to our previous argument, we can see that in an optimal solution, w completely determines g as in equation (1.1). Among all the optimal solutions for (2.1), we pick one (w_0, g_0) maximizing $w(G-T)$, the total weight of

non-tree edges. In any optimal solution, for $e \in G - T$, we claim $w(e) \leq w(T_e)$, because otherwise we could reduce $w(e)$ to $w(T_e)$ while remaining others in w unchanged to get a larger detour gap number $g(G - T)$. Thus,

$$w(G - T) \leq \sum_{e \in G - T} w(T_e) \leq \sum_{e \in G - T} w(T) \leq |E|.$$

Therefore, $w(G - T)$ is bounded, and the point (w_0, g_0) exists.

Lemma 1. *In (w_0, g_0), there does not exist any non-tree edge which is critical to another non-tree edge.*

Otherwise, suppose we have two non-tree edges e and f, and f is critical to e. Then we can increase $w_0(f)$ by δ ($\delta > 0$, δ can be arbitrarily small) while keeping other weights unchanged to get a new weighting w_0' of the graph G. Since $f \notin T$, increasing its weight cannot violate a tree constraint. Now we argue that $g_0'(G - T) \geq g_0(G - T)$.

Compare the two weightings w_0' and w_0. For small enough $\delta > 0$, we have both

$$g_0(f) - g_0'(f) = \delta \qquad \text{and}$$

$$g_0(e) - g_0'(e) = -\delta,$$

while other gaps stay unchanged or increase. Thus, $g_0(G - T) \leq g_0'(G - T)$. Therefore,

$$w_0(G - T) < w_0'(G - T)$$

$$g_0(G - T) \leq g_0'(G - T).$$

This contradicts our choice of w_0 maximizing $w(G - T)$.

Lemma 2. *In (w_0, g_0), $\forall\, e \in G - T$, if $g_0(e) < 0$, then e is not in any shortest detour path of another non-tree edge f.*

Suppose, for some $e \in G - T$, $g_0(e) < 0$. Since f is not critical to e according to Lemma 1, there must exist a shortest detour path P_e of e which does not contain f. Here we have $w_0(P_e) < w_0(e)$ because $g_0(e) < 0$. Thus, a shortest detour path of f should not contain e. Otherwise, we would improve it by replacing e with P_e and shortcutting it to a simple path if needed. (By "shortcutting", we mean eliminating all cycles in a path to get a simple path connecting the same end points.)

Lemma 3. *In an optimal solution (w, g) of (2.1), if $g(e) < 0$, then $w(s) < w(e)$ where s is a tree edge such that $s \triangleleft e$.*

i) If s is on a shortest detour path of e, then $w(s) \leq w(P_e) < w(e)$.

ii) If s is not on any shortest detour path of e, then s crosses at least one non-tree edge f on some shortest detour path P_e of e. Otherwise, e cannot be crossed by s. Thus, $w(s) \leq w(f)$. However, $w(f) \leq w(P_e) < w(e)$ due to $g(e) < 0$, so $w(s) < w(e)$.

Theorem 2. *In (w_0, g_0), there does not exist $e \in G - T$ with $g_0(e) < 0$.*

Proof. Assume that there exists some $e \in G - T$ such that $g_0(e) < 0$. Let's consider a new weighting w_0' of G, in which

$$w_0'(f) = w_0(f), \qquad \forall f \neq e$$
$$w_0'(e) = w_0(e) - \delta.$$

We can pick some small enough $\delta > 0$ such that

$$g_0'(e) > g_0(e) \tag{2.4}$$

$$g_0'(f) = g_0(f), \qquad \forall\, f \in G - T,\ f \neq e^*. \tag{2.5}$$

(2.4) is obvious since the weight of e is reduced while all the weights of its detours remain unchanged. According to Lemma 2, $\forall\, f \in G-T$ and $f \neq e$, e is not in any of its shortest detour paths. So we can certainly pick some δ small enough to still keep e out of the shortest detour flows of all other non-tree edges satisfying (2.5) (A shortest detour flow is the set of all the shortest detour paths of a non-tree edge). Besides, according to Lemma 3, a small enough δ can also keep tree constraints unviolated. Thus, according to (2.4) and (2.5), we have

$$g_0'(G-T) > g_0(G-T),$$

which contradicts with our choice of (w_0, g_0) maximizing $g(G-T)$.

Therefore, the theorem stands. □

Theorem 2 tells us that at least one optimal solution (w_0, g_0) for (2.1) is feasible for (1.3), so

$$gap_0(G,T) \geq rgap_0(G-T).$$

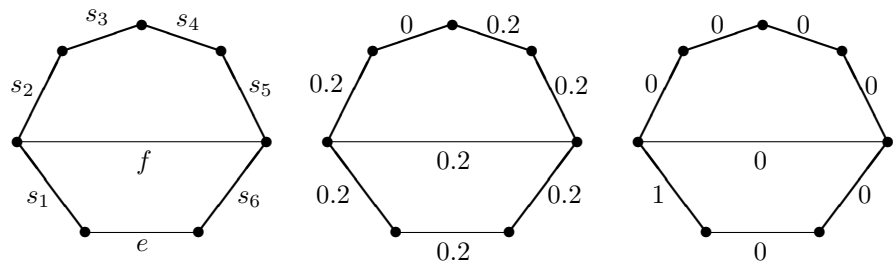
Comparing (2.3), we know

$$gap_0(G,T) = rgap_0(G-T),$$

so linear programs (1.3) and (2.1) are equivalent. A similar proof holds for the equivalence of (1.4) and (2.2) as well. In this case, our argument is simpler because we do not need to worry about preserving the tree constraints.

2.2 Running Example

Now we show an example of the primal problem. Suppose we have a graph G as in Figure 2.1a, where $s_i,\ i = 1, 2, 3, ..., 6$, form the MST T and e, f are two non-tree edges.



(a) (b) $gap_0 = 0.8$ (c) $gap'_0 = 1$

Figure 2.1: Primal Problem

Figure 2.1b gives an optimal solution for linear program (1.3) or (2.1). In Figure 2.1b, for edge e,

$$d_{G-e}(e) = w(s_1) + w(f) + w(s_6) = 0.6,$$

so

$$g(e) = d_{G-e}(e) - w(e) = 0.4.$$

Similarly, we know $g(f) = 0.4$ as well, so for given G and T,

$$gap_0(G,T) = g(G-T) = g(e) + g(f) = 0.8.$$

Moreover, Figure 2.1b also shows $gap_\epsilon(G,T) = 0.8$ for $\epsilon \leq 2$ since it survives the greedy 3-spanner algorithm.

Figure 2.1c gives an optimal solution for linear program (1.4) or (2.2). In Figure 2.1c, for edge e,

$$d_{G-e}(e) = w(s_1) + w(f) + w(s_6) = 1,$$

so

$$g(e) = d_{G-e}(e) - w(e) = 1.$$

Similarly, $g(f) = 0$, so

$$gap'_0(G,T) = g(G-T) = g(e) + g(f) = 1.$$

Chapter 3

Dual Problem

3.1 Charging Scheme

Based on the simplification in Chapter 2, we can use (2.1) to calculate $gap_0(G,T)$. By constructing the dual linear program of (2.1), we can have another interesting view of the problem. We rewrite it in a way to help us construct the dual problem:

$$\begin{aligned}
& gap_0(G,T) = \max_{w,g} \sum_{e \in G-T} g(e) \\
& w(e) \geq 0 && \forall\, e \in G \\
k: \quad & \sum_{e \in T} w(e) \leq 1 \\
t_{s,e}: \quad & w(s) - w(e) \leq 0 && \forall\, e \in G-T, \forall\, s \lhd e \\
c_{e,i}: \quad & g(e) - w(P_{e,i}) + w(e) \leq 0 && \forall\, (P_{e,i}, e) \in \mathcal{P}
\end{aligned} \tag{3.1}$$

Note that each constraint of (3.1) now has a label; these labels will be the variables in our dual program.

From (3.1), we construct the dual linear program as follows:

$$
\begin{aligned}
& & gap_0(G,T) &= \min_{k,t,c} k \\
g(e): & & \sum_i c_{e,i} &= 1 \\
w(s), s \in T: & \quad k + \sum_{e \in G-T, s \lhd e} t_{s,e} - \sum_{f \in G-T} \sum_i \delta(P_{f,i}, s) c_{f,i} &\geq 0 \qquad (3.2) \\
w(e), e \notin T: & \quad \sum_i c_{e,i} - \sum_{s \in T, s \lhd e} t_{s,e} - \sum_{f \in G-T} \sum_i \delta(P_{f,i}, e) c_{f,i} &\geq 0 \\
& & k \geq 0,\ t_{s,e} &\geq 0,\ c_{e,i} \geq 0
\end{aligned}
$$

where for $e \in G$ and $f \in G - T$,

$$
\delta(P_{f,i}, e) = \begin{cases} 1 & e \in P_{f,i} \\ 0 & e \notin P_{f,i} \end{cases}.
$$

Note that in the dual program, each constraint is labeled by a primal variable.

That is

$$
\begin{aligned}
& gap_0(G,T) = \min_{k,t,c} k \\
& \sum_i c_{e,i} = 1 \\
\sum_{f \in G-T} \sum_i \delta(P_{f,i}, s) c_{f,i} - \sum_{e \in G-T, e \lhd s} t_{s,e} &\leq k \qquad s \in T \qquad (3.3) \\
\sum_{f \in G-T} \sum_i \delta(P_{f,i}, e) c_{f,i} + \sum_{s \in T, s \lhd e} t_{s,e} &\leq 1 \qquad e \notin T \\
& k,\ t_{s,e},\ c_{e,i} \geq 0
\end{aligned}
$$

We can think a feasible solution (k, t, c) of (3.3) as describing a charging scheme. We fix G, T and consider t and c as units of charge traded between the edges of G. $t_{s,e}$ units of charge is sent by a tree edge s and received by a non-tree edge e where $s \lhd e$; or we can say $t_{s,e}$ units of charge is charged from s to e. $c_{e,i}$ units of charge is charged from a non-tree edge e to all edges on its ith

detour path $P_{e,i}$. Each edge receives the same amount of charge as its source sends, respectively. That implies the net "charge" on the whole graph is not conserved. In such a charging scheme, we define two types of charging moves.

Definition 2. *A charging move is the action of sending some units of charge from one edge to another edge or to every edge on one of its detour path. There are two types of valid charging moves in the context.*

Type I Charging Move *is the action of sending some units of charge from a tree edge to a non-tree edge it crosses.*

Type II Charging Move *is the action of sending some units of charge from a non-tree edge to all edges on one of its detour paths.*

(Notice that the net "charge" on the graph is not conserved.)

Then a feasible dual solution (k, t, c) can be considered as a series of valid charging moves of the two types.

For all $e \in G - T$, the total charge it sends out is the total charge it sends to its detour paths; that is

$$out(e) = \sum_i c_{e,i}.$$

The total charge it receives is sent by another non-tree edge or a tree edge; that is

$$in(e) = \sum_{f \in G-T} \sum_i \delta(P_{f,i}, e) c_{f,i} + \sum_{s \in T, s \lhd e} t_{s,e}$$

For all $s \in T$, the total charge it sends out can only be received by non-tree edges it crosses; that is

$$out(s) = \sum_{e \in G-T,\ s \lhd e} t_{s,e}$$

The charge it receives can only be sent by some non-tree edge whose detour path contains s; that is

$$in(s) = \sum_{e \in G-T} \sum_{i} \delta(P_{e,i}, s) c_{e,i}$$

From the perspective of sending and receiving charge, the dual linear program (3.3) lays the following constraints on a charging scheme:

$$\begin{aligned} gap_0(G,T) = \min_{k,t,c} & \; k \\ out(e) = 1 \qquad & \forall\, e \in G - T \\ in(e) \leq 1 \qquad & \forall e \in G - T \\ in(s) - out(s) \leq k \qquad & \forall\, s \in T \end{aligned} \tag{3.4}$$

where $in(s) - out(s)$ can also be written as $net(s)$. This is slightly different from what we may find from [8] because we have $out(e) = 1,\ \forall\, e \in G - T$ in our construction, while [8] has it as $out(e) \geq 1,\ \forall\, e \in G - T$. This is the result from simplifying the primal problem by removing the primal constraint $g(e) \geq 0,\ \forall\, e \in G - T$.

Simply speaking, the dual problem turns the calculation of gap_0 to the problem of finding a charging scheme minimizing k, in which each non-tree edge sends out exactly one unit of charge and receives no more than one unit of charge while each tree edge receives no more than k units of net charge.

As for linear problem (2.2), its dual charging scheme forbids tree edges from sending out charge since we don't have t constraints in the primal. Its dual is

simply as follows:

$$gap_0'(G,T) = \min_{k,t,c} k$$

$$out(e) = 1 \qquad \forall\, e \in G - T$$

$$in(e) \leq 1 \qquad \forall e \in G - T$$

$$in(s) \leq k \qquad \forall\, s \in T$$

There are no *Type I* charging moves in this charging scheme, so $out(s) = 0$ for all $s \in T$.

3.2 Running Example

Figure 3.1 is again the example we gave in Chapter 2 (Figure 2.1). Given G as shown in the figure, $T = \{s_1, s_2, s_3, s_4, s_5, s_6\}$ is a spanning tree of G, and e, f are two non-tree edges. Then

$$s_i \triangleleft e,\ i = 1, 2, 3, 4, 5, 6,$$

and

$$s_j \triangleleft f,\ j = 2, 3, 4, 5.$$

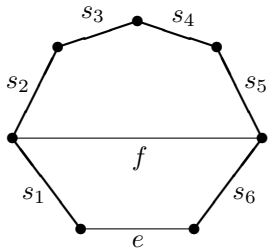


Figure 3.1: Charging Scheme

So the charging scheme on this example is: each s_i charges $t_{s_i,e}$ units of charge to e, and each s_j charges $t_{s_j,f}$ units of charge to f. Besides, e has two detour paths $s_1 f s_6$ and $s_1 s_2 s_3 s_4 s_5 s_6$ while f has two, $s_2 s_3 s_4 s_5$ and $s_1 e s_6$, as

well. Therefore, e charges $c_{e,1}$ units of charge to every edge on the path of $s_1 f s_6$, and charges $c_{e,2}$ to $s_1 s_2 s_3 s_4 s_5 s_6$. Likewise, f charges $c_{f,1}$ to $s_2 s_3 s_4 s_5$ and $c_{f,2}$ to $s_1 e s_6$. Then the linear program (3.4) sets the following constraints for the charging scheme:

$$\begin{aligned}
gap_0(G,T) &= \min_{k,t,c} k \\
c_{e,1} + c_{e,2} &= 1 \\
c_{f,1} + c_{f,2} &= 1 \\
\sum_{i=1}^{6} t_{s_i,e} + c_{f,2} &\le 1 \\
\sum_{j=2}^{5} t_{s_j,f} + c_{e,1} &\le 1 \\
c_{e,1} + c_{e,2} + c_{f,2} - t_{s_m,e} &\le k \qquad m = 1, 6 \\
c_{e,2} + c_{f,1} - t_{s_n,e} - t_{s_n,f} &\le k \qquad n = 2, 3, 4, 5
\end{aligned} \tag{3.5}$$

An optimal solution for (3.5) is

$$\begin{aligned}
gap_0 = k &= 0.8 \\
t_{s_1,e} = t_{s_6,e} &= 0.4 \\
c_{e,1} &= 1 \\
c_{f,1} &= 0.8 \\
c_{f,2} &= 0.2 \\
\text{other } t\text{'s and } c\text{'s} &= 0
\end{aligned} \tag{3.6}$$

It gives exactly the value of gap_0 the primal problem (1.3) gave us before.

If we forbids tree edges from sending out charge, the linear program (3.5) is

now:

$$
\begin{aligned}
gap'_0(G,T) &= \min_{k,t,c} k \\
c_{e,1} + c_{e,2} &= 1 \\
c_{f,1} + c_{f,2} &= 1 \\
c_{f,2} &\leq 1 \\
c_{e,1} &\leq 1 \\
c_{e,1} + c_{e,2} + c_{f,2} &\leq k \qquad m = 1, 6 \\
c_{e,2} + c_{f,1} &\leq k \qquad n = 2, 3, 4, 5
\end{aligned}
\tag{3.7}
$$

An optimal solution for (3.7) is

$$
\begin{aligned}
gap'_0 &= k = 1 \\
c_{e,1} &= 1 \\
c_{f,1} &= 1 \\
\text{other } c\text{'s} &= 0
\end{aligned}
\tag{3.8}
$$

It gives exactly the value of gap'_0 as the primal problem (1.4).

Chapter 4

Combined Primal and Dual

4.1 Complementary Slackness

If we put the primal and dual problems together, we can derive some "complementary slackness" properties of optimal solutions. Now let's consider the charging scheme on a connected simple weighted graph G, in which both weights and charge are what we are interested in.

We fix G and its MST T. Suppose (w, g) is a feasible solution for the primal linear program (3.1) or (2.1) and (k, t, c) is a feasible solution for the dual (3.3) or (3.4).

Each *Type I* move sends $t_{s,e}$ (units of) charge from a tree edge s to a non-tree edge e. Multiplying the charge with edge weight, we have

$$w(e)t_{s,e} - w(s)t_{s,e} = (w(e) - w(s))t_{s,e} \geq 0. \tag{4.1}$$

Each *Type II* move sends $c_{e,i}$ charge from a non-tree edge e to its detour

path $P_{e,i}$. Again, multiplying the charge with edge weight, we have

$$w(P_{e,i})c_{e,i} - w(e)c_{e,i} \geq g(e)c_{e,i}. \tag{4.2}$$

In equations (4.1) and (4.2), the first term of the left hand side is always "charge received times weight", the second term is "charge sent times weight". Since the whole charging scheme is made up of those two types of moves, if we sum up all such equations in (4.1) over all crossing pairs and (4.2) over all $(e, P_{e,i}) \in \mathcal{P}$, we get

$$\sum_{e \in G} w(e)net(e) \geq 0 + \sum_{e \in G-T} \sum_{i} g(e)c_{e,i} = \sum_{e \in G-T} g(e) \sum_{i} c_{e,i} = \sum_{e \in G-T} g(e), \tag{4.3}$$

where $net(e) = in(e) - out(e)$.

On the other hand, since (k, t, c) is feasible for the dual problem (3.3), or (3.4), we know

$$net(e) \leq 0 \qquad \forall\, e \in G - T,$$

$$net(s) \leq k \qquad \forall\, s \in T.$$

So

$$w(e)net(e) \leq 0 \qquad \forall\, e \in G - T, \tag{4.4}$$

$$w(s)net(s) \leq k \cdot w(s) \qquad \forall\, s \in T. \tag{4.5}$$

Besides, since (w, g) is feasible for (3.1), we have

$$w(T) \leq 1.$$

Therefore, (4.3) can be bounded as:

$$\begin{aligned} k &\geq k \cdot \sum_{s \in T} w(s) + 0 \\ &\geq \sum_{s \in T} w(s) net(s) + \sum_{e \in G-T} w(e) net(e) \\ &= \sum_{e \in G} w(e) net(e) \\ &\geq \sum_{e \in G-T} g(e) \\ &= g(G-T). \end{aligned} \tag{4.6}$$

According to strong duality, we know

$$gap_0(G-T) = \min_{k,t,c} k = \max_{w,g} g(G-T);$$

that is, (4.6) is equation when (w, g) is optimal for the primal problem and (k, t, c) optimal for the dual. To achieve equality, we must have equality for all (4.1), (4.2), (4.4) and (4.5). Thus, we have

$$w(s) = w(e) \quad or \quad t_{s,e} = 0 \qquad \forall\, s \lhd e \tag{4.7}$$

$$w(P_{e,i}) - w(e) = g(e) \quad or \quad c_{e,i} = 0 \qquad \forall\, (e, P_{e,i}) \in \mathcal{P} \tag{4.8}$$

$$w(e) = 0 \quad or \quad net(e) = 0 \qquad \forall\, e \in G - T \tag{4.9}$$

$$w(s) = 0 \quad or \quad net(s) = k \qquad \forall\, s \in T \tag{4.10}$$

(4.7) - (4.10) tells us the complementary slackness property of an optimal primal solution (w, g) with an optimal dual solution (k, t, c). For such a bioptimal situation, we conclude:

1. A tree edge can only send charge to a non-tree edge of the same weight;
2. A non-tree edge can only send charge to a shortest detour path;
3. The net charge on a non-tree edge with positive weight is 0;

4. The net charge on a tree edge with positive weight is exactly $k = gap_0$.

As for linear program (2.2), the complementary slackness properties (4.7) - (4.10) are as follows, because leaving out t constraints makes tree edges unable to send charge:

$$w(P_{e,i}) - w(e) = g(e) \quad or \quad c_{e,i} = 0 \quad \forall\, (e, P_{e,i}) \in \mathcal{P}$$

$$w(e) = 0 \quad or \quad net(e) = 0 \quad \forall\, e \in G - T$$

$$w(s) = 0 \quad or \quad net(s) = k \quad \forall\, s \in T$$

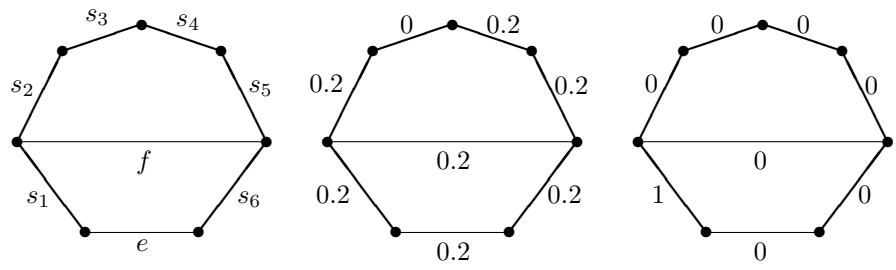
In summary:

1. A non-tree edge can only send charge to a shortest detour path;
2. The net charge on a non-tree edge with positive weight is 0;
3. The net charge on a tree edge with positive weight is exactly $k = gap_0'$.

The constraints above make the charging scheme even simpler, and also without *Type I* charging moves to reallocate charge, there may be more 0-weight edges in an optimal charging solution, which makes the graph modification we are going to discuss in the next chapter more significant and intuitive.

4.2 Running Example

Let's take another look at the example we gave previously. In the following graph, given G and $T = \{s_1, s_2, s_3, s_4, s_5, s_6\}$, the primal solution is shown in Figure 4.1b and 4.1c:



(a) (b) $gap_0 = 0.8$ (c) $gap'_0 = 1$

Figure 4.1: $gap_0(G, T)$

and the dual solution is given as in (3.6) and (3.8):

$$gap_0 = k = 0.8$$

$$t_{s_1,e} = t_{s_6,e} = 0.4$$

$$c_{e,1} = 1$$

$$c_{f,1} = 0.8$$

$$c_{f,2} = 0.2$$

$$\text{other } t\text{'s and } c\text{'s} = 0$$

and

$$gap'_0 = k = 1$$

$$c_{e,1} = 1$$

$$c_{f,1} = 1$$

$$\text{other } c\text{'s} = 0$$

where

$$P_{e,1} = s_1 f s_6$$

$$P_{e,2} = s_1 s_2 s_3 s_4 s_5 s_6$$

$$P_{f,1} = s_2 s_3 s_4 s_5$$

$$P_{f,2} = s_1 e s_6$$

We can verify our conclusion one by one, for Figure 4.1b:

1. s_1 sends 0.4 charge to e since they have the weight of 0.2. The same with s_6 and e.

2. e only charges to its shortest detour path $P_{e,1}$. f charges to $P_{f,1}$ and $P_{f,2}$ since they are both shortest.

3. e and f both have 0 net charge.

4. s_1, s_2, s_4, s_5, s_6 all have 0.8 net charge.

And for Figure 4.1c:

1. e only charges to its shortest detour path $P_{e,1}$. f charges to its shorts, $P_{f,1}$.

2. f has 0 net charge but e does not. However, this doesn't violate our conclusion since $w(e) = 0$.

2. $w(s_1) \neq 0$ and s_1 has 1 net charge.

Chapter 5

Graph Modification

Due to strong duality, in optimal (w, g) and (k, c, t) for (G, T), we see that if the weight of an edge is not equal to 0, then we know the net charge of the edge, either k or 0, according to whether it is a tree edge or not. Now we want to do some modifications to the graph to remove all edges with 0 weight by edge contraction or edge deletion. If we can get a graph minor H of G where there is no 0-weight edges, then we know how the net charge distribution in H looks like in a charging scheme. At the same time, such *non*-0 *minor* H has the same gap_0 as the original graph G. A *non*-0 *minor* is what we would call the basic structure of a graph.

5.1 Tree Edge Contraction

First we contract all tree edges with 0 weight in (G, T) to get a simpler graph (G', T').

If a non-tree edge e disappears due to this operation, then the two end points

of e must be connected by a path containing only tree edges with 0 weight, so e is not on any shortest detour path of other non-tree edges. Thus

$$w(e) = 0$$

Otherwise, we may increase $g(e)$ and $g(G - T)$ by reducing $w(e)$ to 0, which contradicts with the optimality of (w, g). Therefore, $g(e) = w(P_e) - w(e) = 0 - 0 = 0$. Its disappearance won't hurt the total gap of the graph.

If a non-tree edge e remains existence in G', then

$$g(e) = g'(e)$$

where $g'(e)$ is the gap of e in (G', T'). This is because the length of the shortest detour path of e does not change, since only 0 weight edges are contracted.

Accordingly, after tree edge contraction, the total gap of the graph doesn't change. Tree edge contraction applies to both cases with and without tree (t) constraints.

5.2 Parallel Edge Deletion

After tree edge contraction, there is no edge with 0 weight in the graph. However, there may exist parallel edges connecting two end points. Now we introduce a way to delete parallel edges for the special case of (w_0, g_0). Obviously, two tree edges won't be parallel to each other. Otherwise, tree edges must form a cycle before contraction. Here we consider the following two cases.

i) A tree edge s is parallel to a non-tree edge e.

In this case, $w_0(s) \leq w_0(e)$ due to tree constraints. On the other hand, $w_0(s) - w_0(e) \geq g_0(e) \geq 0$, so $w_0(s) = w_0(e)$ and $g_0(e) = 0$. Thus, removing e makes no difference to the total gap of the graph.

ii) A non-tree edge e is parallel to another non-tree edge f while no tree edge is parallel to them.

Similar to the argument above, we can prove $w_0(e) = w_0(f)$. Now we introduce the definition of an edge group.

Definition 3. *An edge group is the set of all (≥ 2) parallel non-tree edges connecting the same two end points.*

When we say "increasing" or "decreasing" the weight of an edge group, we mean "increasing" or "decreasing" the weight of every edge in that edge group by the same amount, since we don't want to violate the non-negative gap property of (w_0, g_0).

Then we can restate Lemma 1 for an edge group.

Lemma 4. *In (w_0, g_0), there does not exist any edge group which is critical to a non-tree edge. By "critical", we mean the gap of the non-tree edge will increase when we increase the weight of the edge group.*

Proof. It is easy to know that every edge in an edge group has a gap of 0. Suppose we have an edge group M and a non-tree edge $e \notin M$, and M is critical to e. Then we can increase the weight of M by δ ($\delta > 0$, δ can be any small) while keeping other weights in the graph unchanged to get a new weighting w_0'. Since $M \not\subset T$, increasing its weight cannot violate a tree constraint. Now we argue that $g_0'(G - T) \geq g_0(G - T)$.

Compare the two weightings w_0' and w_0. Since δ can be any small, we can always find such a δ that

$$g_0(e) - g_0'(e) = \delta$$

$$g_0(M) = g_0'(M) = 0$$

while other gaps stay unchanged or increase. Thus, $g_0(G-T) < g'_0(G-T)$. Therefore,

$$w_0(G-T) < w'_0(G-T)$$

$$g_0(G-T) < g'_0(G-T)$$

This contradicts with our choice of (w_0, g_0) maximizing $w(G-T)$.

Therefore, we can always remove an edge group from the graph without changing the total gap. Note that parallel edge deletion only applies to the case with tree constraints. □

After tree edge contraction and parallel edge deletion, we get a *non*-0 *minor* H of graph G where there is no 0-weight edges and no parallel edges. H is the basic structure determining the gap of the graph. Such *non*-0 *minor* would be worth further research.

Chapter 6

Nonmonotonicity in Edge Deletion

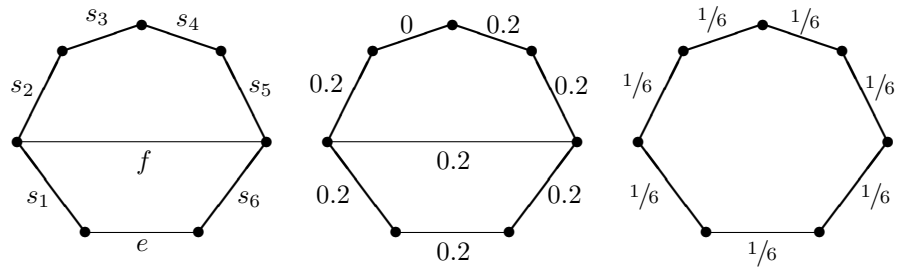
Definition 4. *A charging scheme is acyclic if there is some total ordering of the edges such that whenever e charges f, e precedes f in the ordering.*

It is easy to see that if a charging scheme is acyclic, then we can delete an edge and get another acyclic scheme of the same gap value or smaller. If a graph has an acyclic optimal charging scheme, then its gap is monotone under edge deletion. Then it raises such a question: is detour gap monotone under edge deletion for all graphs? This question is the same as: can we always find an acyclic optimal charging scheme for a given graph?

However, we have a counterexample to show gap_0 is not monotone. It is still the same example we used through this thesis:

Figure 6.1b is the problem we have seen previously. Its gap is

$$gap_{0b} = 0.8.$$



(a) (b) $gap_{0b} = 0.8$ (c) $gap_{0c} = 0.833$

Figure 6.1: Nonmonotonicity of gap_0

After deleting the edge of f, the calculation result as shown in Figure 6.1c is that

$$gap_{0c} = 5/6 = 0.833 > gap_{0b}.$$

Thus, gap_0 increases after edge deletion. That means no acyclic optimal charging scheme exists for this graph and gap_0 is not monotone under edge deletion. Moreover, the same example also shows that gap_ϵ is not monotone under edge deletion for $\epsilon \leq 2$.

However, as for the case leaving out tree constraints, our example does not violate the monotonicity under edge deletion. That leaves an open question: can we always find an *acyclic* optimal charging scheme under the constraints of gap'_0, where only *Type II* moves are valid in a charging scheme?

Chapter 7

Software

To facilitate the analysis of detour gap numbers, we developed a software to calculate $gap_0(G,T)$ and $gap'_0(G,T)$ for a given graph G and its spanning tree T. The software is developed in Java using a linear program solver library *lp_solve* to solve the following two linear programs to get $gap_0(G,T)$ and $gap'_0(G,T)$:

$$
\begin{aligned}
gap_0(G,T) = \max_{w,g,d_{G-e}} \; & g(G-T) \\
w(e) &\geq 0 && \forall\, e \in G \\
g(e) &\geq 0 && \forall\, e \in G \\
d_{G-e}(u,x) &\geq 0 && \forall\, e=(u,v) \in G,\ x \in V \\
w(T) &\leq 1 \\
w(s) &\leq w(e) && \forall\, e \in G-T,\ \forall\, s \lhd e \\
g(e) &\leq d_{G-e}(e) - w(e) && \forall\, e \in G-T \\
d_{G-e}(u,u) &\leq 0 && \forall\, u \in V \\
d_{G-e}(u,y) &\leq d_{G-e}(u,x) + w(x,y) && \forall\, e=(u,v) \in G-T, \\
& && \forall\, (x,y) \in G-e
\end{aligned}
\tag{7.1}
$$

$$
\begin{aligned}
gap'_0(G,T) = \max_{w,g,d_{G-e}} \; & g(G-T) \\
w(e) &\geq 0 && \forall\, e \in G \\
g(e) &\geq 0 && \forall\, e \in G \\
d_{G-e}(u,x) &\geq 0 && \forall\, e=(u,v) \in G,\ x \in V \\
w(T) &\leq 1 \\
g(e) &\leq d_{G-e}(e) - w(e) && \forall\, e \in G-T \\
d_{G-e}(u,u) &\leq 0 && \forall\, u \in V \\
d_{G-e}(u,y) &\leq d_{G-e}(u,x) + w(x,y) && \forall\, e=(u,v) \in G-T, \\
& && \forall\, (x,y) \in G-e
\end{aligned}
\tag{7.2}
$$

Comparing (1.3) and (7.1), we introduced the metric variable d_{G-e} to reduce

the exponentially many constraints:

$$g(e) \leq w(P_{e,i}) - w(e) \qquad \forall\ (P_{e,i}, e) \in \mathcal{P}$$

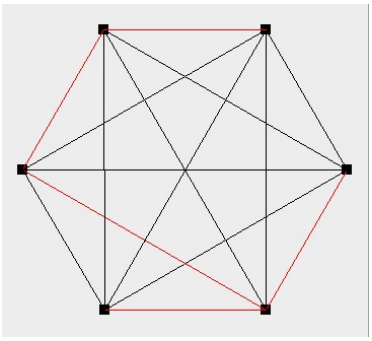
to a polynomial number. Note that for each edge $e = (u, v) \in G - T$, the optimal value of $d_{G-e}(u, x)$ is the length of a shortest path from u to x in $G - e$. In Václav Chvátal's book [10], there are more detailed explanations on this typical type of linear program in finding the shortest paths in a given graph. d_{G-e} can be considered as the cost of reaching out from the source point, which increments after each step further toward the target. It is similar with (1.4) and (7.2).

The software has an interactive user interface for inputing a graph and its spanning tree by drawing each point and edge, or users may input a graph and a tree by an input file. The software uses a graph library *jgrapht* to deal with the simple weighted graph. According to the graph specified and the linear program introduced above, the software generates an LP object for the primal problem, either for (7.1) or for (7.2). By solving it, the software outputs an optimal solution along with its dual value for each constraint. Based on the solution, it is also easy to query the shortest detour flow of a chosen non-tree edge. By selecting a non-tree edge, the software can highlight its detour flow in bright color with different line width proportional to the charge on each edge.

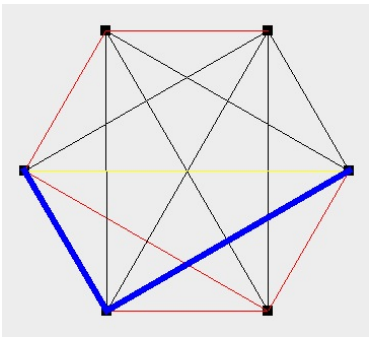
The source code is available on the URL:

http://www.mathcs.emory.edu/∼slou3/Gap/.

Here are screen shots of the software:



(a) Graph and MST



(b) Detour Path (Flow)

Figure 7.1: Primal Problem

Bibliography

[1] G. Gutin, A.P. Punnen. The Traveling Salesman Problem and Its Variations. *Volume 12 of Combinatorial Optimization*. Springer, 2002. ISBN 1402006640, 9781402006647.

[2] Richard M. Karp. Reducibility Among Combinatorial Problems. In R. E. Miller and J. W. Thatcher (editors). *Complexity of Computer Computations*, pages 85-103. New York: Plenum, 1972.

[3] S. Arora, M. Grigni, D. Karger, P. Klein, and A. Woloszyn. A polynomial-time approximation scheme for weighted planar graph TSP. In *Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 33-41, 1998.

[4] M. Grigni, E. Koutsoupias, and C. Papadimitriou. An approximation scheme for planar graph TSP. In *28th Annual Symposium on Foundations of Computer Science*, pages 640-646. IEEE, Oct. 1995.

[5] Brenda S. Baker. Approximation algorithms for NP-complete problems on planar graphs. *Journal of the Association for Computing Machinery*, 41(1):153-180, 1994.

[6] André Berger, Artur Czumaj, Michelangelo Grigni, and Hairong Zhao. Approximation schemes for minimum 2-connected spanning subgraphs in weighted planar graphs. In *Proceedings of the 13th Annual European Symposium on Algorithms*, volume 3669 of *Lecture Notes in Computer Science*, pages 472-483. Palma de Mallorca, Spain, October 2005.

[7] I. Althöfer, G. Das, D. P. Dobkin, D. Joseph, and J. Soares. On sparse spanners of weighted graphs. *Discrete Comput. Geom.*, 9(1):81-100, 1993.

[8] Michelangelo Grigni and Papa Amar Sissokho. Light spanners and approximate TSP in weighted graphs with forbidden minors. In *Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 852-857, 2002.

[9] M. Grigni. Approximate TSP in graphs with forbidden minors. In *Automata, Languages and Programming: 27th International Colloquium*, volume 510 of *Lecture Notes in Computer Science*, pages 869-877. Springer-Verlag, July 2000.

[10] Václav Chvátal. Linear Programming, pages 291-320. W.H. Freeman. ISBN 0-7167-1195-8, 1983.