

Algorithms for the minimum non-separating path and the balanced connected bipartition problems on grid graphs (With errata)

Bang Ye Wu*

Dept. of Computer Science and Information Engineering
National Chung Cheng University, Taiwan

Abstract

For given a pair of nodes in a graph, the minimum non-separating path problem looks for a minimum weight path between the two nodes such that the remaining graph after removing the path is still connected. The balanced connected bipartition (BCP_2) problem looks for a way to bipartition a graph into two connected subgraphs with their weights as equal as possible. In this paper we present an algorithm in time $O(N \log N)$ for finding a minimum weight non-separating path between two given nodes in a grid graph of N nodes with positive weight. This result leads to a $5/4$ -approximation algorithm for the BCP_2 problem on grid graphs, which is the currently best ratio achieved in polynomial time. We also developed an exact algorithm for the BCP_2 problem on grid graphs. Based on the exact algorithm and a rounding technique, we show an approximation scheme, which is a fully polynomial time approximation scheme for fixed number of rows.

Key words. algorithm, approximation algorithm, non-separating path, balanced connected partition, grid graphs.

About this version

This article was published as [16]. In this version we report a mistake about the *minimum non-separating path* on *grid graphs* and fix the $5/4$ -approximation algorithm for *Balanced Connected 2-Partition* problem on grid graphs ($GBCP_2$). The correction is at the appendix.

1 Introduction

Let $G = (V, E, w)$ be a connected undirected graph, in which w is a nonnegative node weight function. For two given nodes s and t , if we want to allocate some of the nodes for the communication between s and t , choosing a minimum st -path (a minimum weight path with endpoints s and t) may be the best way. However, if the chosen nodes cannot be used for other services, the remaining network may be separated into species. To

*National Chung Cheng University, ChiaYi, Taiwan 621, R.O.C., E-mail: bangye@cs.ccu.edu.tw

keep the remaining network connected, one may hope to find a minimum *non-separating st-path*, i.e., a *st-path* P such that the remaining graph $G - P$ is connected. However, in general, non-separating path does not always exist. A natural relaxation allows any connected subgraph containing both s and t . That is, we look for a minimum weight connected subgraph B containing both s and t such that $G - B$ is connected. We name such a subgraph by “non-separating *st-connector*” (*st-NSC* or simply “NSC” if no confusion), and the one with minimum weight is a *minimum non-separating connector* (min-NSC).

A node bipartition $(U, V - U)$ is a *connected bipartition* if both the subgraphs induced by U and $V - U$ are connected. Immediately if B is an NSC, then $(V(B), V - V(B))$ is a connected bipartition. The maximum balance connected bipartition (BCP₂) problem looks for a connected bipartition $(U, V - U)$ such that the balance, defined by $\min\{w(U), w(V - U)\}$, is maximized, in which $w(U)$ denotes the total weight of nodes in U . The applications of BCP may appear in image processing, data bases, operating systems, cluster analysis, etc. [5]. An $m \times n$ grid graph M is an undirected graph and can be thought of as a 2-dimensional matrix, in which m and n are the numbers of rows and columns, respectively. The node set of M can be represented by $V = \{M_{ij} | 1 \leq i \leq m, 1 \leq j \leq n\}$ and there exists an edge between two consecutive nodes in the same row or the same column. In this paper we study the min-NSC and the BCP₂ problems on node-weighted grid graphs.

The non-separating path problem has been studied from the perspective of graph theory. Most of the works are devoted to its relationship to graph connectivity [4, 6, 9, 10, 13], but we haven’t found any optimization problem about it. The min-NSC problem on general graphs is NP-hard in the strong sense and cannot be approximated with ratio $|V|^{1-\epsilon}$ for any $\epsilon > 0$ in polynomial time unless $P=NP$ [15] (named “minimum border problem”). In this paper we show that a minimum *st-NSC* on a grid graph is a minimum non-separating *st-path* and can be found in $O(N \log N)$ time, in which N is the number of nodes. The efficient algorithm is based on two key points. First, the min-NSC on a grid graph is a minimum weight path with at most one boundary subpath; and secondly, such a path can be found by reducing to a *range minimum query* (RMQ) problem.

The second result of this paper is about the BCP₂ problem on grid graphs (GBCP₂ for short). Based on NSC and *st-numbering*, we propose a 5/4-approximation algorithm with time complexity $O(N \log N)$ for the GBCP₂ problem, which is the currently best result achieved in polynomial time. We also developed an exact algorithm for GBCP₂. For an $m \times n$ grid graph of total weight W , $m \leq n$, the algorithm takes $O(mNW8^m)$ time, which is more efficient than the naive brute force method of $O(N2^N)$ time. The exact algorithm uses a typical dynamic programming strategy and computes the best bipartition for any possible weight and any connection topology of the first i columns for i from 1 to n . The analysis itself is of its own interest. An obvious upper bound of the number of connection topologies is m^m . With a more precise analysis and using the method of generating function, we show a sharper bound of $O(2^m)$. Based on the exact algorithm and a rounding technique, we developed an approximation scheme. For any $\epsilon > 0$, the GBCP₂ problem can be $(1 + \epsilon)$ -approximated in $O((1 + \frac{1}{\epsilon})mN^28^m)$ time, which is a *fully polynomial-time approximation scheme* (FPTAS) for fixed m .

The BCP _{q} problem is a generalization of BCP₂, for which the input graph is partitioned into q connected subgraphs for any given $q \geq 2$. Previous results about BCP₂ are as follows. The BCP₂ on grid graphs of more than two rows was shown to be NP-hard

[1] while for grid graphs of two rows (also known as “ladders”), the problem can be solved in polynomial time [2]. Approximation algorithms for BCP_q on grid graphs were also presented by [1] but the general approximation ratios were not given, except for the case $q = 2$, for which a $3/2$ -approximation can be guaranteed. Besides the ladders, it is known that the BCP_q problem is polynomially solvable for trees [12] and unweighted q -connected graphs [11]. [7] showed that BCP_2 on general graphs is NP-hard in the strong sense and cannot be approximated with an absolute error guarantee of $|V|^{1-\varepsilon}$ for any $\varepsilon > 0$ unless $\text{NP}=\text{P}$. A $4/3$ -approximation algorithm was also given in that paper, which is currently the best approximation ratio of the problem, even on grid graphs. For BCP_3 and BCP_4 , on 3- and 4-connected graphs respectively, there are 2-approximation algorithms proposed by [5].

The rest of the paper is organized as follows: In Section 2, we give some notations and show that the min-NSC is a minimum non-separating path in a grid graph. In Section 3, we show the algorithm for the minimum non-separating path. The $5/4$ -approximation algorithm for GBCP_2 is given in Section 4. The exact algorithm and the approximation scheme of GBCP_2 are in Sections 5 and 6, respectively. Finally, some concluding remarks are given in Section 7.

2 Preliminaries

Let $G = (V, E)$ be a graph, $S \subset V$ and H an induced subgraph of G . The subgraph induced by S is denoted by $G[S]$. By $G - S$ we denote $G[V - S]$, and similarly $G - H = G[V - V(H)]$, in which $V(H)$ is the node set of H . Let $w : V \rightarrow \mathbb{Z}^+$ be a node weight function. By $w(S)$, we denote the total weight of S , i.e., $w(S) = \sum_{v \in S} w(v)$. For convenience $w(H) = w(V(H))$. Let $[i, j]$ denote the interval of integers $\{i, i + 1, \dots, j\}$ for $i \leq j$. An $m \times n$ grid graph M will be thought of as an $m \times n$ matrix such that $V(M) = \{M_{ij} | i \in [1, m], j \in [1, n]\}$ and there exists an edge between two consecutive nodes in the same row or the same column. The set of nodes in the first or the last row and the first or the last column is called as the *boundary* of the grid graph. The four nodes $M_{1,1}$, $M_{1,n}$, $M_{m,1}$ and $M_{m,n}$ are called as corner nodes. W.l.o.g. we assume $n \geq m$. Let $N = mn = |V(M)|$ and w_{ij} the weight of node M_{ij} .

Let \mathcal{C}_1 , \mathcal{C}_2 and \mathcal{C}_3 be the sets of all non-separating induced st -paths, non-separating st -paths and st -NSCs, respectively. By definition, $\mathcal{C}_1 \subset \mathcal{C}_2 \subset \mathcal{C}_3$. For general graphs, they are different and there may be even no any non-separating st -path. Figure 1 illustrates a case that the minimum NSC and the non-separating (induced) st -path are different. In the remaining of this section we show that $\mathcal{C}_1 = \mathcal{C}_2 = \mathcal{C}_3$ on a grid graph except that $\{s, t\}$ is a *2-cut*.

A node subset is a *cut* if the graph becomes disconnected after its removal. A 2-cut is a cut consisting of two nodes. For a grid graph of at least three rows, the only 2-cuts are the pairs of the two neighbors of corner nodes. Suppose that $\{s, t\}$ is a 2-cut of an $m \times n$ grid graph G , in which $n \geq m \geq 3$. Let x be the corner node adjacent to both s and t . Apparently the minimum st -NSC is either the path (s, x, t) or $G - x$, depending on which weight is smaller. The minimum non-separating path is similar but a little tricky. It is not hard to observe that the minimum non-separating st -path is either (s, x, t) or a Hamiltonian st -path of $G - x$ if the Hamiltonian path exists. If both m and n are odd integers, we can show that there does not exist a Hamiltonian st -path in $G - x$ as follows. First we color s white, and then all other nodes are colored according to the

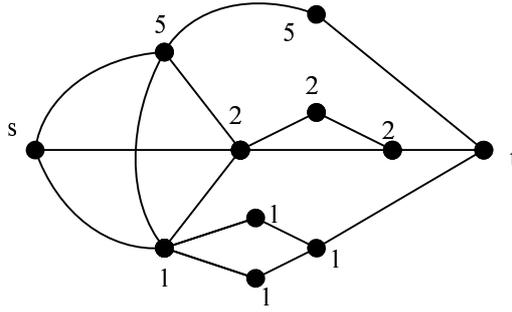


Figure 1: The minimum NSC and the minimum non-separating (induced) st -path. The nodes are labeled by their weights. The minimum st -NSC consists of the four nodes of weight 1 (in addition to s and t); the minimum non-separating st -path passes through the three nodes of weight 2; and the minimum non-separating induced st -path passes through the two nodes of weight 5.

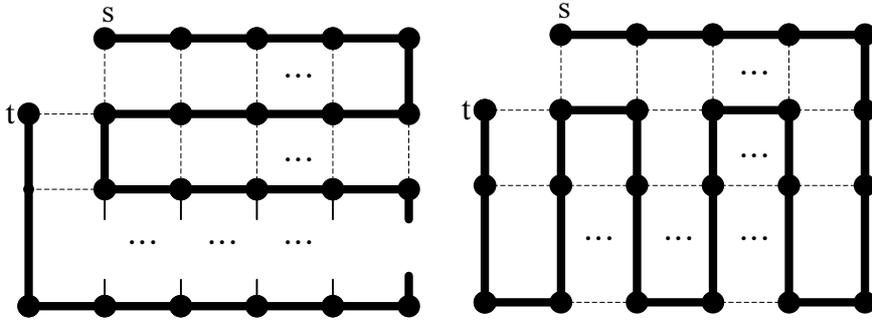


Figure 2: The Hamiltonian st -paths in $G - x$ for even number of rows and even number of columns are shown in the left and the right, respectively

rule: “the neighbors of a white node should be color black and vice versa”. It can be easily checked that t is colored white and that the numbers of white and black nodes are the same. Since a path with both endpoints colored white cannot have equal number of white and black nodes, no Hamiltonian st -path exists on $G - x$. On the contrary, if m or n is even, we can show that a Hamiltonian st -path always exists. W.l.o.g we assume that x is the corner node $M_{1,1}$. The Hamiltonian paths for even number of rows, and columns respectively, are illustrated in Fig. 2.

Anyway, if $\{s, t\}$ is a 2-cut, both the min-NSC and the minimum non-separating path can be easily computed, and we shall assume it is not the case in the remaining of the paper. For an NSC B or a connected partition $(V(B), V - V(B))$, a node v is movable if it is still a connected bipartition after moving v to the other part. A trivial observation is that $v \in V(B)$ is movable iff v is not an articulation node in B and v has a neighbor in $V - V(B)$, assuming $V - V(B)$ is not empty. For a connected bipartition of a biconnected graph, there are at least two movable nodes in each part unless the part contains less than two nodes [7]. The following result comes from the minimality of the NSC.

Lemma 1: If B is a minimum st -NSC of a biconnected graph, both s and t are movable

and there is no other movable node in B .

Theorem 2: If $\{s, t\}$ is not a 2-cut, a minimum st -NSC B on a grid graph M of at least three rows is a non-separating induced st -path.

Proof: By definition $M - B$ is connected, and it is sufficient to show that B is an induced st -path. A block is either a maximally biconnected subgraph or a bridge (an edge whose removal disconnects the graph). Let \mathcal{K} and A be the sets of the blocks and the cut vertices of B , respectively. The *block-cutpoint tree* T of B is defined as follows [14, Chap. 4]. $V(T) = \mathcal{K} \cup A$, and for any $a \in A$ and $K \in \mathcal{K}$, $(a, K) \in E(T)$ iff, in the original graph B , a is a vertex in block K . By definition T is a tree and each leaf of T corresponds to a block of B .

Since M is biconnected, for any leaf K of T , there is a node in the block K of B which is adjacent to $M - B$ and therefore movable. By Lemma 1, s and t are the only movable vertices in B . So T has at most two leaves, and is therefore a path. We shall show that each block is an edge, and the proof is completed. Suppose by contradiction that K is a block of B and $|V(K)| > 2$. Since M is a grid graph of at least three rows, there are at least three nodes in K adjacent to $M - K$ unless K contains all nodes but a corner (in this case $K = B$). If K contained all nodes but a corner node, then s and t would be the two neighbors of the corner node and therefore a 2-cut, violating the assumption. A node in K and adjacent to $M - K$ is either

- a cut of B and therefore adjacent to a component of $B - K$; or
- adjacent to $M - B$ and therefore a movable node of B because no node of a block of more than two nodes is a cut of the block.

Since M is biconnected, there is a movable node of B in each component of $B - K$. Therefore there are at least three movable nodes in B , and by Lemma 1, B is not a minimum st -NSC. \square

Corollary 3: For a grid graph of at least three rows, the minimum NSC, the minimum non-separating path and the minimum non-separating induced path are all equivalent except that s and t are the two neighbors of a corner node.

3 Minimum non-separating path

In this section we show how to find a minimum non-separating path efficiently. A boundary path is a path along the boundary, i.e., all the nodes are on the boundary. A interior path is a path whose internal nodes are not on the boundary. In the following we shall denote by \mathcal{B} the boundary of M . For a path P , a *boundary subpath* of P is a *maximal* subpath of P with all nodes on the boundary, i.e. a boundary subpath of P is not a subpath of another boundary subpath of P . A subpath may contain only a single node.

Lemma 4: A non-separating induced st -path has at most one boundary subpath.

Proof: If P is an induced path and has more than one boundary subpaths, there are at least two boundary segments divided by P , and each of these segments is in one component of $M - P$, which implies P is not non-separating. \square

Lemma 5: If P is a minimum st -path with at most one boundary subpath, then P is a non-separating induced path.

Proof: Let $P = (s = v_1, v_2, \dots, v_l = t)$. First we show that P is an induced path by contradiction. If not, there exist v_i and v_j such that $(v_i, v_j) \in E$ and $i < j - 2$. Since removing a subpath from a path will not increase the number of boundary subpaths, we can replace the subpath from v_i to v_j with the edge (v_i, v_j) to obtain a path of less weight and with at most one boundary subpath, a contradiction to the minimality of P .

For any induced path, the subgraph induced by any of its node subset is not a cycle. Therefore P cannot include the whole boundary. Since P has at most one boundary subpath, the remaining boundary nodes $\mathcal{B} - V(P)$ are connected and not empty. If P was not non-separating, there would be an interior node separated from the remaining boundary by P . Since there is no induced cycle in $V(P)$ and P only has at most one boundary subpath, it only happens at the case that the remaining boundary contains only a corner node, which implies that P is a boundary path including the whole boundary but the corner node, i.e., s and t are the two neighbors of the corner node. But this contradicts the assumption that $\{s, t\}$ is not a 2-cut. \square

By the above results, a minimum NSC is a minimum non-separating path, and is also a minimum st -path with at most one boundary subpath. This property is helpful for designing our algorithm. The next corollary is immediate.

Corollary 6: If both s and t are on the boundary, a minimum non-separating st -path is a boundary path and can be found in linear time to $|\mathcal{B}|$.

For two nodes u and v , let $d_B(u, v)$, and $d_I(u, v)$, denote the minimum weight of any boundary path, and interior path respectively, between u and v . For $d_B(u, v)$, both u and v must be on the boundary. Let $d(u, v)$ denote the minimum weight of any non-separating induced uv -path. We can have the next lemma.

Lemma 7: If s is on the boundary and t is not, $d(s, t) = \min_{i \in \mathcal{B}} \{d_B(s, i) + d_I(i, t) - w(i)\}$.

Proof: By Lemma 5, it is sufficient to compute the minimum weight of any st -path with at most one boundary subpath. Since s is on the boundary, the optimal path must be a concatenation of a boundary subpath and an interior subpath, including the degenerating case that the boundary subpath is only one node, i.e., s in this case. \square

For each u of the four corner nodes, the value $d_I(v, u) = \infty$ for any v . It is trivial that the total time complexity to compute $d_B(s, i)$ for every $i \in \mathcal{B}$ is linear to $|\mathcal{B}|$. To compute $d(s, t)$, it is sufficient to find $d_I(t, i)$ for every $i \in \mathcal{B}$, and then the minimum can be found in $O(|\mathcal{B}|)$ time. Since any boundary node i other than a corner node has only one interior neighbor, $d_I(t, i)$ is the minimum weight of any path between t and the interior neighbor of i on $M - \mathcal{B}$. Since a minimum weight path can be found by an algorithm similar to Dijkstra's algorithm and the number of edges in a grid graph is linear to the number of nodes, the time complexity is dominated by calling to Dijkstra algorithm, which takes time $O(N \log N)$ [8].

Corollary 8: If s is on the boundary and t is not, then $d(s, t)$ can be found in time $O(N \log N)$.

The proof of the next result is similar to Lemma 7 and is omitted.

Lemma 9: If neither s nor t is on the boundary, then $d(s, t)$ is the minimum between $d_I(s, t)$ and

$$\min_{i, j \in \mathcal{B}} \{d_I(s, i) + d_B(i, j) + d_I(j, t) - w(\{i, j\})\} \quad (1)$$

Note that if the optimal path contains a boundary node, it must contain at least two boundary nodes since a boundary node has at most one interior neighbor. The time for computing $d_I(s, t)$ is $O(N \log N)$ by Dijkstra algorithm. To compute (1), a naive algorithm of checking all possible i and j takes quadric time. We shall give an algorithm with time complexity $O(|\mathcal{B}|)$ in the following.

Starting at an arbitrary boundary node, we number the boundary nodes clockwise from 1 to $|\mathcal{B}|$. Let $w(i)$ be the weight of $i \in \mathcal{B}$ and W_B denote the total weight of boundary nodes. Our algorithm has two rounds. In the first round, we find for every i the best j such that the minimum boundary path from i to j is clockwise. The other case that the path is counterclockwise is checked in the second round. Since the two rounds are similar, we shall only show the first round. For convenience, we double the whole sequence, i.e., the $(|\mathcal{B}| + i)$ -th node is the same as the i -th node for $1 \leq i \leq |\mathcal{B}|$. The minimum boundary ij -path is clockwise if

$$\sum_{i < k < j} w(k) \leq \frac{1}{2}(W_B - w(i) - w(j)).$$

Therefore we define $\text{right}(i)$ as the maximum index j in $[i + 1, i + |\mathcal{B}| - 1]$ such that the path is clockwise.

Since $\text{right}(i)$ is an increasing function, it is not hard to show that computing $\text{right}(i)$ for every i can be done in total time $O(|\mathcal{B}|)$. For every i from 1 to $|\mathcal{B}|$, we find $j^* \in (i, \text{right}(i)]$ minimizing $d_I(s, i) + \sum_{i < k < j^*} w(k) + d_I(t, j^*)$. For a fixed i , equivalently we only need to find j^* minimizing $\sigma(j^*) = \sum_{1 \leq k < j^*} w(k) + d_I(t, j^*)$. By this way we reduce our problem to a *range minimum query* (RMQ) problem. For a one dimensional array, there is an algorithm which reports the minimum in any index range in constant time after a linear-time preprocessing [3]. For our problem, the array σ has $2|\mathcal{B}|$ elements and we need to perform $|\mathcal{B}|$ queries. Therefore the time complexity for finding indices i and j minimizing (1) is $O(|\mathcal{B}|)$. Clearly $\sigma(j)$ for every j can be found in $O(N \log N)$ time. Combining this result with Corollaries 6 and 7, we summarize this section in the next theorem.

Theorem 10: A minimum non-separating path on an N -nodes grid graph can be found in $O(N \log N)$ time.

4 A 5/4-approximation algorithm for GBCP_2

In this section we show a 5/4-approximation algorithm with time complexity $O(N \log N)$ for the GBCP_2 problem by using minimum non-separating paths. For the GBCP_2 problem, the optimal solution is trivial if there exists a node of weight at least $W/2$. Therefore we shall exclude this case in the following. We shall first introduce an algorithm for finding a bipartition of any biconnected graph but not necessarily of a grid graph. This

algorithm is based on st -numbering and finds a $4/3$ -approximation of BCP_2 . Then we show how to improve the ratio to $5/4$ for grid graphs by using minimum non-separating paths.

4.1 An algorithm based on st -numbering

For a biconnected undirected graph $G = (V, E)$ and $s, t \in V$, an st -numbering is a 1-to-1 labeling $\lambda : V \rightarrow [1, |V|]$ satisfying $\lambda(s) = 1$, $\lambda(t) = |V|$; and, for each node $v \in V - \{s, t\}$, v has a neighbor with label smaller than $\lambda(v)$ and also a neighbor with label larger than $\lambda(v)$. For a biconnected graph, an st -numbering always exists and can be found in linear time [Even and Tarjan(1976)]. The original algorithm for st -numbering requires that s and t must be adjacent. But, if $(s, t) \notin E$, we can simply add edge (s, t) to obtain an st -numbering.

Algorithm STN

Input: A biconnected graph $G = (V, E)$.

Output: A connected bipartition G .

- 1: find two nodes s and t of the largest and the second largest weights, respectively;
 - 2: compute an st -numbering λ ;
 - 3: let $V = \{v_i | 1 \leq i \leq n\}$ such that $\lambda(v_i) = i, \forall i$;
 - 4: find k such that $w(V_k) \leq W/2$ and $w(V_{k+1}) > W/2$, in which $V_k = \{v_i | 1 \leq i \leq k\}$;
 - 5: **if** $w(V_k) \geq W - w(V_{k+1})$ **then**
 - 6: $k^* \leftarrow k$;
 - 7: **else**
 - 8: $k^* \leftarrow k + 1$;
 - 9: **end if**
 - 10: output $(V_{k^*}, V - V_{k^*})$.
-

Claim 11: For any $1 \leq k < n$, the bipartition $(V_k, V - V_k)$ is a connected bipartition.

Proof: For any node $v \in V_k$, since each node has a neighbor with smaller st -number, there exists a path from v to s in $G[V_k]$. Consequently $G[V_k]$ is connected. It can be shown similarly that $G[V - V_k]$ is also connected. \square

Lemma 12: The algorithm STN takes linear time. If V_{k^*} is the solution produced by the algorithm, then $\min\{w(V_{k^*}), W - w(V_{k^*})\} \geq (W - w_3)/2$, where w_3 is the third largest node weight in G .

Proof: Due to [Even and Tarjan(1976)], the st -numbering can be computed in linear time. It is trivial that all the other steps can also be done in linear time.

When $k^* = k$, i.e., $w(V_k) \geq W - w(V_{k+1})$, we have $\min\{w(V_{k^*}), W - w(V_{k^*})\} = w(V_k) \geq W - w(V_{k+1})$. Similarly, when $k^* = k + 1$, i.e., $w(V_k) < W - w(V_{k+1})$, we have $\min\{w(V_{k^*}), W - w(V_{k^*})\} = W - w(V_{k+1}) > w(V_k)$. That is, in either case, we have $\min\{w(V_{k^*}), W - w(V_{k^*})\} \geq \max\{w(V_k), W - w(V_{k+1})\} \geq (w(V_k) + (W - w(V_{k+1}))) / 2 = (W - w(v_{k+1})) / 2$. Since no node has weight larger than $W/2$, we have that v_{k+1} is neither s nor t . Therefore $w(v_{k+1}) \leq w_3$. \square

4.2 A 5/4-approximation algorithm

By Lemma 12 and the analysis in [7], it can be shown that the algorithm STN is a linear-time 4/3-approximation algorithm for the BCP₂ problem on any biconnected graph. Of course it works for grid graphs since a grid graph is biconnected. The remaining paragraphs of this section aims at improving the approximation ratio to 5/4. In the remaining we assume G is an $m \times n$ grid graph and $N = m \times n$, in which $n \geq m \geq 3$. Let $H = \{h_i | w(h_i) > W/5\}$ be the set of *heavy* nodes. Clearly $|H| \leq 4$. We shall show how to find a 5/4-approximation solution for each possible value of $|H|$. The minimum non-separating path will play an important role in the case of $|H| = 3$.

Claim 13: When $|H| \leq 2$, the algorithm STN finds a 5/4-approximation.

Proof: In this case the third largest node weight w_3 is at most $W/5$. By Lemma 12, the returned bipartition satisfies $\min\{w(V_{k^*}), W - w(V_{k^*})\} \geq (W - W/5)/2 = (2/5)W$. Comparing with the trivial upper bound $W/2$, the ratio is 5/4. \square

Claim 14: When $|H| = 4$, the algorithm STN finds a 5/4-approximation.

Proof: Since the nodes are arranged in a linear order by their *st*-numbers, there always exists a bipartition, said \mathcal{P} , whose both parts contain exactly two heavy nodes and are of weight larger than $(2/5)W$. By the optimality of k^* , the output is no worse than \mathcal{P} . \square

Finally we consider the case that $|H| = 3$.

Definition 1: Let G be a graph and U a subset of nodes of G . The contracted graph G/U is the graph obtained by combining all the nodes in U by a new node u and, for any $v \notin U$, the edge (u, v) exists iff v has a neighbor in U . For convenience, $G/S = G/V(S)$ for a subgraph S .

Lemma 15: If G is biconnected and $(U, V(G) - U)$ is a connected bipartition, then G/U is biconnected.

Proof: Since $(U, V(G) - U)$ is a connected bipartition, the new node u is not a cut node in G/U . Since G is originally biconnected, no node in $V(G) - U$ will be a cut node in the contracted graph.

Our 5/4-approximation algorithm for the case of three heavy nodes is as follows.

Algorithm THREE_HEAVY

- 1: let $H = \{h_1, h_2, h_3\}$ be the set of nodes of weight larger than $W/5$;
 - 2: find a minimum $h_i h_j$ -NSC B_{ij} for each $1 \leq i < j \leq 3$;
 - 3: let B be the NSC of smallest weight among those found in Step 2;
 - 4: **if** $w(B) < W/2$ **then**
 - 5: call algorithm STN on the contracted graph G/B ;
 - 6: **else**
 - 7: output $(V(B), V - V(B))$.
 - 8: **end if**
-

Claim 16: In the case of $|H| = 3$, the algorithm `THREE_HEAVY` finds a $5/4$ -approximation in $O(N \log N)$ time.

Proof: By Lemma 15, the contracted graph G/B is biconnected and has exactly two nodes of weight larger than $W/5$. If $w(B) < W/2$, the result is the same as the case of $|H| = 2$. For otherwise we claim that $(V(B), V - V(B))$ is an optimal connected bipartition. Let $(U, V - U)$ be an optimal connected bipartition. Since there are three heavy nodes, at least two heavy nodes must be in the same part. W.l.o.g. we assume that heavy nodes h_1 and h_2 are in U . By the definitions of NSC and B , we have that $w(U) \geq w(B_{12}) \geq w(B) \geq W/2$, and this implies $(V(B), V - V(B))$ is an optimal connected bipartition.

By the result of previous section, B must be an induced path or the whole grid graph lacking a corner node. And in either case the contracted graph G/B can be easily constructed in linear time. Since the algorithm `STN` takes also linear time, the total time complexity is dominated by the step of finding the minimum NSC's, which is $O(N \log N)$ according to Theorem 10. □

We conclude this section as follows.

Theorem 17: The GBCP_2 can be approximated with ratio $5/4$ in $O(N \log N)$ time.

5 An exact algorithm for GBCP_2

In this section we develop an exact algorithm for GBCP_2 . We shall assume that the grid graph has at least three rows. For $1 \leq j \leq n$ and a bipartition (V_0, V_1) , we use a vector $\mathbf{z}_j \in \{0, 1\}^m$ to represent a bipartition of the j -th column such that $\mathbf{z}_j^i = 0$ if $M_{ij} \in V_0$ and is one otherwise, in which \mathbf{z}_j^i denotes the i -th component of \mathbf{z}_j for $1 \leq i \leq m$.

We shall represent by configuration $(\mathbf{z}, \theta, \tau)$ a possible bipartition (V_0, V_1) of the first j columns such that the partition of the j -th column is \mathbf{z} , the weight of V_1 is θ , and τ represents how the nodes of column j are connected in the first j columns. That is, for the first j columns, if we delete the nodes of V_1 , τ represents the connected components of the nodes of V_0 ; and similarly the components of V_1 if we delete the nodes of V_0 . We say τ is a *connection topology*. We shall develop a dynamic programming algorithm computing all possible configurations for j from 1 to n . We first discuss the connection topology.

5.1 Connection topology

To represent a connection topology, it is sufficient to use a data structure for disjoint sets. Each set contains the row indices of the nodes in the same connected component. Precisely speaking, at column j , we use an array τ such that $\tau[i]$ stores the representer of the set M_{ij} belongs to, in which the representer of a set is the smallest row index of its members. For example, if, at the first j columns, the nodes $M_{2,j}$, $M_{3,j}$ and $M_{9,j}$ are in the same component, we record the set $\{2, 3, 9\}$ by $\tau[2] = \tau[3] = \tau[9] = 2$. Since the nodes in a component are all in V_0 or V_1 , we divide a connection topology into a 0's connection topology, or 0-topology for short, and a 1-topology. That is, a q -topology is a partition of $\{i | \mathbf{z}_j^i = q\}$ for column j . A trivial upper bound of the number of connection

topologies is m^m , and we aim at finding a much sharper bound. We shall first consider the 0-topology.

First of all, if two adjacent nodes are in V_0 , they must be in the same subset. We transform a column vector into a set of disjoint segments of 0's (0-segments for short). Labeling the segments by consecutive integers from 1, we have an interval $[1, p]$. Clearly $p \leq \lceil m/2 \rceil$. For the above example of $\{2, 3, 9\}$, 2 and 3 are the first segment and 9 is the second segment. We shall represent the bound by a function of p . In the remaining a 0-topology shall be thought of as a partition of $[1, p]$.

Consider the points $\{(i, 0) | 1 \leq i \leq p\}$ in the Euclidean plane. The number of 0-topologies is the number of ways of adding some lines on one half-plane to joint some of these points. If two lines are cross, the corresponding segments are in the same component.

Definition 2: For two subsets S_1 and S_2 in a 0-topology \mathcal{T} , S_1 is *covered* by S_2 if $\max(S_2) > \max(S_1)$ and $\min(S_2) < \min(S_1)$, in which $\max(S_1)$ and $\min(S_1)$ denote the maximum and the minimum elements in S_1 , respectively. A subset is *covered* if it is covered by some other subset and is *uncovered* otherwise.

A crucial observation is as follows:

If \mathcal{T} is a 0-topology of $[1, i]$ and $i \in S \in \mathcal{T}$, then $S - \{i\}$ cannot be covered by any $S' \in \mathcal{T}$.

Let $\alpha(i, j)$ denote the number of 0-topologies of i segments with j uncovered subsets for $1 \leq j \leq i \leq p$. In any 0-topology of i segments, i must be in one of the uncovered subsets of a 0-topology of $i - 1$ segments. For a 0-topology of $i - 1$ segments with k uncovered subsets, joining i to the j -th uncovered subsets will form a 0-topology of i segments with j uncovered subsets, for $j \leq k \leq i - 1$. Besides, leaving i itself as an uncovered subset along with a $j - 1$ uncovered subsets also forms a 0-topology of j uncovered subsets, seeing Fig. 3.(a). Therefore we can have the following recurrence relation.

$$\alpha(i, j) = \sum_{k=j-1}^{i-1} \alpha(i-1, k) \quad \text{for } 1 \leq j < i \quad (2)$$

The boundary conditions are $\alpha(i, 0) = 0$ and $\alpha(i, i) = 1$ for any i .

Lemma 18: For a fixed 0-topology with j uncovered subsets, there are at most 2^j 1-topologies.

Proof: Clearly the 0-segments interleave the 1-segments, and vice versa. Let \mathcal{T} be a 0-topology. For $S \in \mathcal{T}$, we say that a 1-segment i is *minimally covered* by S iff i is covered by S but not covered by any $S' \in \mathcal{T}$ covered by S . As shown in Fig. 3.(b), for any $S \in \mathcal{T}$, all the 1-segments minimally covered by S is in the same component. So, for the 1-segments covered by at least one subsets in \mathcal{T} , their components are fixed. Therefore the number of 1-topologies only depends on the uncovered 1-segments. If there are j uncovered subsets in S , there are at most $j + 1$ uncovered 1-segments. Let $f(k)$ denote the maximum number of 1-topologies with k uncovered 1-segments. Clearly $f(1) = 1$. For $k > 1$, the k -th 1-segment either forms a component by itself or joins to the $(k - 1)$ -th

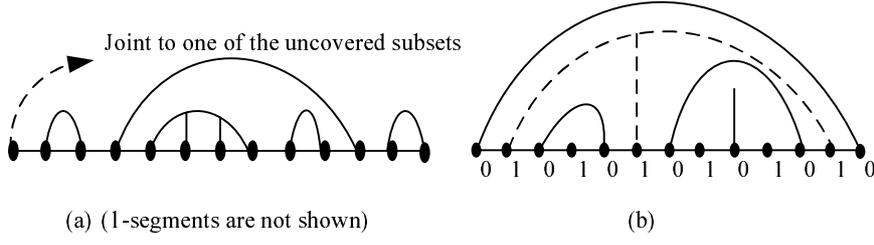


Figure 3: Connection topologies (the columns are transposed).

1-segment. Note that since they are uncovered, it cannot join to other segment but skip from the $(k-1)$ -th one. Consequently we have $f(k) = 2f(k-1)$ for $k > 1$, and it turns out to be $f(k) = 2^{k-1} \leq 2^j$. \square

Corollary 19: For p 0-segments, the number of connection topologies is upper bounded by $t(p) = \sum_{1 \leq j \leq p} 2^j \alpha(p, j)$.

Lemma 20: For any $h \geq 0$ and positive i , $\sum_{k=0}^h \binom{i-1+k}{i-1} \leq \binom{i+h}{i}$.

Proof: By induction on h . First when $h = 0$, $\sum_{k=0}^h \binom{i-1+k}{i-1} = \binom{i-1}{i-1} = \binom{i+h}{i}$. Suppose that the inequality holds for $h-1$, i.e., $\sum_{k=0}^{h-1} \binom{i-1+k}{i-1} \leq \binom{i+h-1}{i}$. We have

$$\begin{aligned} \sum_{k=0}^h \binom{i-1+k}{i-1} &= \sum_{k=0}^{h-1} \binom{i-1+k}{i-1} + \binom{i-1+h}{i-1} \\ &\leq \binom{i+h-1}{i} + \binom{i+h-1}{i-1} = \binom{i+h}{i} \end{aligned}$$

\square

Lemma 21: $\alpha(i, j) \leq \binom{2^{i-j}}{i}$ for $1 \leq j \leq i$.

Proof: By induction on i . When $i = 1$, the only feasible value of j is 1, and $\alpha(1, 1) = 1 = \binom{2^{1-1}}{1}$. Suppose that it holds for $i-1$ and any $1 \leq j \leq i-1$. By (2),

$$\alpha(i, j) = \sum_{k=j-1}^{i-1} \alpha(i-1, k) \leq \sum_{k=j-1}^{i-1} \binom{2^{(i-1)-k}}{i-1} = \sum_{k=0}^{i-j} \binom{i-1+k}{i-1}$$

The last equality is obtained by substituting k by $i-1-k$. Then, by Lemma 20, $\alpha(i, j) \leq \binom{2^{i-j}}{i}$. \square

Lemma 22: $t(p) \leq 2^{2^p} - \binom{2^p}{p}$ for $p \geq 1$.

Proof: By Corollary 19 and Lemma 21,

$$t(p) = \sum_{1 \leq j \leq p} 2^j \alpha(p, j) \leq \sum_{1 \leq j \leq p} 2^j \binom{2p-j}{p} = \sum_{j=0}^{p-1} 2^{p-j} \binom{p+j}{j} \equiv \hat{t}(p)$$

The last equality is obtained by substituting j with $p-j$. We derive a closed form of $\hat{t}(p)$ by the method of generating function. First, $2^{p-j} \binom{p+j}{j}$ is the coefficient of x^j in $2^{-j}(2+x)^{p+j}$, and is the coefficient of x^p in $2^{-j}x^{p-j}(2+x)^{p+j}$. Then $\hat{t}(p)$ is the coefficient of x^p in the following generating function:

$$T(x) = \sum_{j=0}^{p-1} 2^{-j} x^{p-j} (2+x)^{p+j} \quad (3)$$

Since

$$\begin{aligned} \frac{2x}{2+x} T(x) &= \sum_{j=0}^{p-1} 2^{-j+1} x^{p-j+1} (2+x)^{p+j-1} \\ &= \sum_{j=-1}^{p-2} 2^{-j} x^{p-j} (2+x)^{p+j} \end{aligned} \quad (4)$$

subtracting (4) from (3) and solving $T(x)$, we obtain

$$\begin{aligned} T(x) &= \left(\frac{1}{2-x} \right) (2^{1-p} x (2+x)^{2p} - 2x^{p+1} (2+x)^p) \\ &= \left(\sum_{i=0}^{\infty} \left(\frac{x}{2} \right)^i \right) (2^{-p} x (2+x)^{2p} - x^{p+1} (2+x)^p) \end{aligned}$$

Since $\hat{t}(p)$ is the coefficient of x^p in $T(x)$, the second term is useless, and

$$\begin{aligned} \hat{t}(p) &= 2^{-p} \sum_{i=0}^{p-1} \left(\frac{1}{2} \right)^i \binom{2p}{p-i-1} 2^{p+i+1} = 2 \sum_{i=0}^{p-1} \binom{2p}{p-i-1} \\ &= 2 \sum_{i=0}^{p-1} \binom{2p}{i} = \sum_{i=0}^{2p} \binom{2p}{i} - \binom{2p}{p} = 2^{2p} - \binom{2p}{p} \end{aligned}$$

□

Since p is the number of 0-segments and bounded by $\lceil m/2 \rceil$, we obtain the following result.

Theorem 23: The number of connection topologies is $O(2^m)$.

5.2 An exact algorithm for GBCP_2

Let L_j be the list of all configurations for column j . Initially, for each binary m -vector \mathbf{z} , there is only one connection topology τ and only one possible total weight θ of V_1 . So L_1 contains 2^m configurations. For j from 2 to n , our algorithm computes L_j from

L_{j-1} in each iteration, as well as checks if a better feasible solution is obtained. We say a component is “closed” at column $j - 1$ if the component contains at least one node of column $j - 1$ but none of column j . For any $(\mathbf{z}, \theta, \tau) \in L_{j-1}$ and any bipartition \mathbf{z}' of column j , we check the following conditions:

- If $V_q, q \in \{0, 1\}$, has exactly one component in $(\mathbf{z}, \theta, \tau)$:
 - all nodes of column $\geq j$ assigned to V_{1-q} is a feasible solution. We need only check if it is the currently best solution but not insert it into L_j .
 - any other \mathbf{z}' closing the component is illegal and should be discarded.
- Otherwise any \mathbf{z}' closing any component at column $j - 1$ is illegal.
- If it is not illegal, compute θ' and τ' , and insert $(\mathbf{z}', \theta', \tau')$ into L_j .

Finally for each configuration of column n , we check if it is feasible and update the best one if necessary. We next analyze the time complexity.

To store the configurations, we use a $2^m \times W$ table T and each entry of the table is a list of connection topologies. For a configuration $(\mathbf{z}, \theta, \tau)$, we store τ in the list of $T[\mathbf{z}, \theta]$. By using a balanced binary search tree (such as AVL tree) with array τ as the key, we can check the existence and insert a configuration in $O(m^2)$ time. Note that we need to check and avoid the duplicates to ensure the number of configurations is bounded. The number of configurations is $O(2^m \times W \times 2^m) = O(W \times 4^m)$. Since there are n iterations and in each iteration it takes $O(m^2)$ time for every configuration and every \mathbf{z}' , the total time complexity is $O(m^2 n W 8^m) = O(m N W 8^m)$. It is much better than $O(N 2^N)$ of the brute force algorithm.

Theorem 24: The GBCP_2 can be solved in $O(m N W 8^m)$, in which N is the number of nodes, m is the number of rows, and W is the total weight.

6 Approximating GBCP_2

Based on the exact algorithm for GBCP_2 and a scaling technique, we shall show an approximation algorithm for GBCP_2 , which is a FPTAS for fixed number of rows.

For some $\rho < 1$, we scale down the weights by a factor $r = \rho W / (3N)$, i.e., we set $w'_{ij} = \lfloor w_{ij} / r \rfloor$ for each i and j . Then we run the exact algorithm on the modified weight w' to obtain a bipartition (V_0, V_1) as the approximation solution. Let (V_0^*, V_1^*) be the optimal bipartition w.r.t. w . The following result can be derived from Lemma 12.

Lemma 25: If no node has weight larger than $W/2$, then $\min\{w(V_0^*), w(V_1^*)\} \geq W/3$.

Lemma 26: $\frac{\min\{w(V_0^*), w(V_1^*)\}}{\min\{w(V_0), w(V_1)\}} \leq \frac{1}{(1-\rho)}$.

Proof: W.l.o.g. let $w(V_0) \leq w(V_1)$. By definition, $w(V_0) \geq r w'(V_0)$. Since (V_0, V_1) is the optimal bipartition w.r.t. w' , $\min_q w'(V_q) \geq \min_q w'(V_q^*)$ and therefore

$$w(V_0) \geq r w'(V_0) \geq r \min_q w'(V_q^*)$$

By the definition of w' , for $q = 0$ or 1 , $w(V_q^*) < rw'(V_q^*) + rN$. Hence, $r \min_q w'(V_q^*) > \min_q w(V_q^*) - rN$, and we obtain

$$w(V_0) > \min_q w(V_q^*) - rN$$

Then, since $r = \rho W/(3N)$ and $\min_q w(V_q^*) \geq W/3$ by Lemma 25, the approximation ratio can be calculated by

$$\min_q w(V_q^*)/w(V_0) \leq \frac{\min_q w(V_q^*)}{\min_q w(V_q^*) - rN} \leq \frac{W/3}{W/3 - \rho W/3} = \frac{1}{1 - \rho},$$

which completes the proof. \square

Theorem 27: For any $\varepsilon > 0$, a $(1 + \varepsilon)$ -approximation of the GBCP_2 can be found in $O((1 + \frac{1}{\varepsilon})mN^28^m)$ time, in which N is the number of nodes and m is the number of rows. For fixed number of rows, it is a FPTAS.

Proof: For any given $\varepsilon > 0$, we set $\rho = \frac{\varepsilon}{1+\varepsilon}$ and $r = \rho W/(3N)$. By Lemma 26, the approximation ratio is $1 + \varepsilon$. By Theorem 24, the time complexity is $O(mNW'8^m) = O(mN(\frac{W}{\rho W/(3N)})8^m) = O((1 + \frac{1}{\varepsilon})mN^28^m)$. \square

7 Concluding remarks

The technique in Section 4 may be used for other classes of graphs. For any graph class on which the minimum non-separating path problem can be solved in polynomial time, the BCP_2 problem can be approximated with ratio $5/4$ in the same time complexity.

The approximation algorithm shown in Section 6 is an FPTAS only for fixed number of rows. The interesting open problems include how to design an FPTAS or PTAS for the GBCP_2 of non-fixed number of rows and how to evenly partition a (grid) graph into more than two connected subgraphs.

Acknowledgements

This work was supported in part by NSC 97-2221-E-194-064-MY3 and NSC 98-2221-E-194-027-MY3 from the National Science Council, Taiwan.

References

- [1] Becker R, Lari I, Lucertini M, Simeone B (1998) Max-min partitioning of grid graphs into connected components. *Networks* 32:115–125
- [2] Becker R, Lari I, Lucertini M, Simeone B (2001) A polynomial-time algorithm for max-min partitioning of ladders. *Theory Comput Syst* 34:353–374

- [3] Bender MA, Farach-Colton M, Pemmasani G, Skiena G, Sumazin P (2005) Lowest common ancestors in trees and directed acyclic graphs. *J Algorithms* 57:75–94
- [4] Bollobás B, Thomason A (1996) Highly linked graphs. *Comb* 16(3):313–320
- [5] Chataigner F, Salgado LRB, Wakabayashi Y (2007) Approximation and Inapproximability results on balanced connected partitions of graphs. *Discret Math and Theor Comput Sci* 9:177–192
- [6] Chen G, Gould RJ, Yu X (2003) Graph connectivity after path removal. *Comb* 23(2):185–203
- [7] Chlebíková J (1996) Approximating the maximally balanced connected partition problem in graphs. *Inf Process Lett* 60:225–230
- [Even and Tarjan(1976)] Even S, Tarjan RE (1976) Computing an st -numbering. *Theor Comput Sci* 2:339–344
- [8] Fredman ML, Tarjan RE (1987) Fibonacci heaps and their uses in improved network optimization algorithms. *J ACM* 34:209–221
- [9] Kawarabayashi KI, Lee O, Yu X (2005) Non-separating paths in 4-Connected graphs. *Ann Comb* 9:47–56
- [10] Kawarabayashi KI, Lee O, Reed B, Wollan P (2008) A weaker version of Lovász’s path removal conjecture. *J Comb Theor, Series B* 98:972–979
- [11] Lovász L (1977) A homology theory for spanning trees of a graph. *Acta Math Acad Sci Hunger* 30:241–251
- [12] Perl Y, Schach S (1981) Max-min tree partitioning. *J ACM* 28(1):5–15
- [13] Tutte WT (1963) How to draw a graph. *Proc London Math Soc* 13:747–767
- [14] West DB (2001) *Introduction to Graph Theory*. Prentice Hall
- [15] Wu BY, Chen HC (2009) The approximability of the minimum border problem. In: *Proceedings of the 26th Workshop on Combinatorial Mathematics and Computation Theory, Taiwan*
- [16] B.Y. Wu, Algorithms for the minimum non-separating path and the balanced connected bipartition problems on grid graphs, *Journal of Combinatorial Optimization*, 26 (2013): 592–607.

Appendix: Erratum

The mistake appears in the proof of Lemma 5 which states that

If P is a minimum st -path with at most one boundary subpath, then P is a non-separating induced path.

and $q \equiv w(Q) - w_3 > 3W/5 - w_3$. W.l.o.g. we assume that $w(P_2) \leq w(P_3)$, and we claim $D = (P_2, G - P_2)$ is a $5/4$ -approximation solution.

If $w(P_2) \leq W/2$, then we have done. Otherwise, the cost of the 2-partition $c(D)$ is

$$\begin{aligned}
W - w(P_2) &\geq q + (W - q - w_2 - w_3)/2 \\
&= W/2 + q/2 - (w_2 + w_3)/2 \\
&> W/2 + 3W/10 - w_3/2 - (w_2 + w_3)/2 \\
&\geq 4W/5 - (w_2 + w_3)/4 - (w_2 + w_3)/2 \\
&= 4W/5 - 3(w_2 + w_3)/4.
\end{aligned}$$

We divide the proof into two cases: $w_2 + w_3 \leq W/2$ or $w_2 + w_3 > W/2$. When $w_2 + w_3 \leq W/2$, we have that

$$c(D) \geq (4/5 - 3/8)W = 17W/40 > 2W/5,$$

and the approximation ratio is less than $5/4$. When $w_2 + w_3 > W/2$, the optimal cost is at most $W - (w_2 + w_3)$, and the ratio is

$$\frac{W - (w_2 + w_3)}{4W/5 - 3(w_2 + w_3)/4} < 20/17 < 5/4.$$