Network Optimization on Partitioned Pairs of Points

Esther M. Arkin¹, Aritra Banik², Paz Carmi², Gui Citovsky¹, Su Jia¹, Matthew J. Katz², Tyler Mayer¹, and Joseph S. B. Mitchell¹

¹ Dept. of Applied Mathematics and Statistics, Stony Brook University.
{esther.arkin, gui.citovsky, su.jia, tyler.mayer, joseph.mitchell}@stonybrook.edu

² Dept. of Computer Science, Ben-Gurion University, Israel.
{aritrabanik, carmi2}@gmail.com, matya@cs.bgu.ac.il

Abstract

Given $n$ pairs of points, $S = \{\{p_1, q_1\}, \{p_2, q_2\}, \ldots, \{p_n, q_n\}\}$, in some metric space, we partition the points in $S = \bigcup_{i=1}^{n} \{p_i, q_i\}$ into two sets, $S_1$ and $S_2$, so that $p_i \in S_1$ if and only if $q_i \in S_2$. The partition should optimize the cost over a pair of network structures, one built on $S_1$ and one built on $S_2$. In this paper we consider our network structures to be spanning trees, traveling salesman tours or matchings. We consider several different weight functions computed over the network structures induced, as well as several different objective functions. We show that some of these problems are NP-Hard, provide constant factor approximation algorithms in all cases.

Digital Object Identifier 10.4230/LIPIcs...

1 Introduction

We study a class of network optimization problems on pairs of sites in a metric space. Our goal is to determine how to split each pair, into a “red” site and a “blue” site, in order to optimize both a network on the red sites and a network on the blue sites. In more detail, given $n$ pairs of points, $S = \{\{p_1, q_1\}, \{p_2, q_2\}, \ldots, \{p_n, q_n\}\}$, in the Euclidean plane or in a general metric space, we partition the points in $S = \bigcup_{i=1}^{n} \{p_i, q_i\}$ into two sets, $S_1$ and $S_2$, so that $p_i \in S_1$ if and only if $q_i \in S_2$. The partition should optimize the cost of certain structures built on both $S_1$ and $S_2$: spanning trees, traveling salesman tours (TSP tours) or matchings. Let $f(X)$ be a certain structure computed on point set $X$ and let $\lambda(X)$ be the bottleneck edge (longest edge) of a structure computed on point set $X$. For each of the aforementioned structures we consider the objective of minimizing $|f(S_1)| + |f(S_2)|$, minimizing $\max\{|f(S_1)|, |f(S_2)|\}$ and minimizing $\max\{|\lambda(S_1)|, |\lambda(S_2)|\}$. Here, $|\cdot|$ denotes the cost (e.g., sum of edge lengths) of the structure.

The problems we study are natural variants of well-studied network optimization problems. Our motivation comes also from a model of secure connectivity in networks involving facilities with replicated data. Consider a set of facilities each having two (or more) replications of their data; the facilities are associated with pairs of points (or $k$-tuples of points in the case of higher levels of replication). Our goal may be to compute two networks (a “red” network and a “blue” network) to interconnect the facilities, each network visiting exactly one data site from each facility; for communication connectivity, we would require each network to be a tree, while for servicing facilities with a mobile agent, we would require each network to be a Hamiltonian path/cycle. By keeping the red and blue networks distinct, a malicious attack within one network is isolated from the other.
Our results.

We show that several of these problems are NP-Hard and give $O(1)$-approximation algorithms for each of them. Table 1 summarizes our $O(1)$-approximation results.

| Problem        | $\min |f(S_1)| + |f(S_2)|$ | $\min\max\{\min|\lambda(S_1)|, |\lambda(S_2)|\}$ | $\min\max\{\min|f(S_1)|, |f(S_2)|\}$ |
|----------------|-----------------------------|---------------------------------|---------------------------------|
| Spanning tree  | 3$\alpha$                   | 9                               | 4$\alpha$                       |
| Matching       | 2                           | 3                               | 3                               |
| TSP tour       | $3\beta$                    | 18                              | $6\beta$                        |

| Table 1 Table of results: $\alpha$ is the Steiner ratio and $\beta$ the best approximation factor of the TSP in the underlying metric space. Unless specified otherwise, all other results in this table apply to general metric spaces. |

Related work.

Several optimization problems have been studied of the following sort: Given sets of tuples of points (in a Euclidean space or a general metric space), select exactly one point or at least one point from each tuple in order to optimize a specified objective function on the selected set. Gabow et al. [9] explored the problem in which one is given a directed acyclic graph with a source node $s$ and a terminal node $t$ and a set of $k$ pairs of nodes, where the objective was to determine if there exists a path from $s$ to $t$ that uses at most one node from each pair. Myung et al. [14] introduced the Generalized Minimum Spanning Tree Problem: Given an undirected graph with the nodes partitioned into subsets, compute a minimum spanning tree that uses exactly one point from each subset. They show that this problem is NP-hard and that no constant-factor approximation algorithm exists for this problem unless $P = NP$. Related work addresses the generalized traveling salesperson problem [16, 15, 5, 17], in which a tour must visit one point from each of the given subsets. Arkin et al. [3] studied the problem in which one is given a set $V$ and a set of subsets of $V$, and one wants to select at least one element from each subset in order to minimize the diameter of the chosen set. They also considered maximizing the minimum distance between any two elements of the chosen set. In another recent paper, Consuegra et al. [7] consider several problems of this kind. Abellanas et al. [1], Das et al. [8], and Khantemouri et al. [12] considered the following problem. Given colored points in the Euclidean plane, find the smallest region of a certain type (e.g., strip, axis-parallel square, etc.) that encloses at least one point from each color. Barba et al. [4] studied the problem in which one is given a set of colored points (of $t$ different colors) in the Euclidean plane and a vector $c = (c_1, c_2, \ldots, c_t)$, and the goal is to find a region (axis-aligned rectangle, square, disk) that encloses exactly $c_i$ points of color $i$ for each $i$. Efficient algorithms are given for deciding whether or not such a region exists for a given $c$.

While optimization problems of the “one of a set” flavor have been studied extensively, the problems we study here are fundamentally different: we care not just about a single structure (e.g., network) that makes the best “one of a set” choices on, say, pairs of points; we must consider also the cost of a second network on the “leftover” points (one from each pair) not chosen. As far as we know, the problem of partitioning points from pairs into two sets in order to optimize objective functions on both sets has not been extensively studied. One recent work of Arkin et al. [2] does address optimizing objectives on both sets: Given a set of pairs of points in the Euclidean plane, color the points red and blue so that if one point of a pair is colored red (resp. blue), the other must be colored blue (resp. red). The objective is to optimize the radii of the minimum enclosing disk of the red points and the minimum enclosing disk of the blue points. They studied the objectives of minimizing the
sum of the two radii and minimizing the maximum radius.

2 Spanning Trees

Let $\text{MST}(X)$ be a minimum spanning tree over the point set $X$, and $|\text{MST}(X)|$ be the cost of the tree. Let $\lambda(X)$ be the bottleneck edge in a spanning tree on point set $X$ and $|\lambda(X)|$ be the cost of the bottleneck edge. Given $n$ pairs of points in a metric space, partition the point set $S$ into two sets, $S_1$ and $S_2$, so that for pair $\{p_i, q_i\}$, $p_i \in S_1$ if and only if $q_i \in S_2$ and the cost of a spanning tree built over each set is minimized.

2.1 Minimum Sum

In this section we consider minimizing $|\text{MST}(S_1)| + |\text{MST}(S_2)|$.

**Theorem 1.** The Min-Sum 2-MST problem is NP-Hard in general metric spaces.

**Proof.** The reduction is from Max 2SAT where one is given $n$ variables $\{x_1, x_2, \ldots, x_n\}$ and $m$ clauses $\{c_1, c_2, \ldots, c_m\}$. Each clause contains at most two literals joined by a logical or. The objective is to maximize the number of clauses that evaluate to true.

For each variable $x_i$ we create a variable gadget that consists of two pairs of points: $\{p_{2i}, q_{2i}\}$ and $\{p_{2i+1}, q_{2i+1}\}$ (see Figure 1). Setting $x_i$ to true is equivalent to using edges $(p_{2i+1}, q_{2i})$ and $(p_{2i}, q_{2i+1})$. Setting $x_i$ to false is equivalent to using edges $(p_{2i}, p_{2i+1})$ and $(q_{2i+1}, q_{2i})$.

Variable gadgets will be arranged on a line with distance $O(L)$ between consecutive variable gadgets for $L = n + m$ (see Figure 2).

For every pair of variable gadgets corresponding to variables $x_i, x_j, i \neq j$ we place a cluster $A_{i,j}$ of $M = m^2$ points near point $p_{2i+1}$. Each of these points is paired to a point in a cluster $B_{i,j}$ of $M$ points near point $q_{2j+1}$. Any two points in the same cluster, $A_{i,j}$ or $B_{i,j}$, are separated by distance two from each other and by distance one from point $p_{2i+1}, q_{2j+1}$ respectively. Note that this enforces points $p_{2i+1}$ and $q_{2j+1}$ to be in different trees for all $1 \leq i, j \leq n$. Otherwise, if $p_{2i+1}$ and $q_{2j+1}$ were placed in the same tree, then connecting the points in clusters $A_{i,j}, B_{i,j}$ to the trees would cost at least $M$ more than it would to have $p_{2i+1}$ and $q_{2j+1}$ in different trees.

Now we argue that the optimal solution uses edges $(p_{2i+1}, p_{2i+3})$ and $(q_{2i+1}, q_{2i+3})$, $1 \leq i \leq n - 1$, as “backbones” of the two MSTs. To see this, observe that if any other edge was used to connect two consecutive variable gadgets, then we would need to use at least one edge of length $L + 2$. Since $p_{2i+1}$ and $q_{2j+1}$ will be in different trees for all $1 \leq i, j \leq n$ and since points $p_{2i}$ and $q_{2j}$ will be connected to points $p_{2i+1}$ and $q_{2i+1}$ ($1 \leq i \leq n$), we have a set of “lower” components that must be connected and a set of “upper” components that need to be connected. No upper component can be connected to a lower component. Any edge of length at least $L + 2$ connecting any of these components can thus be replaced by an edge of length $L$.

The remaining variable gadget points, $\{p_{2i}, q_{2i}\}$ ($1 \leq i \leq n$), must be connected to the backbones. That is, for variable $x_i$, points $p_{2i}$ and $q_{2i}$ will be picked up either by using edges $(p_{2i+1}, q_{2i})$ and $(p_{2i}, q_{2i+1})$ (green in Figure 3) or edges $(p_{2i}, p_{2i+1})$ and $(q_{2i+1}, q_{2i})$ (red in Figure 3). As mentioned, the green edges correspond to setting $x_i$ to true and the red edges correspond to setting $x_i$ to false.

A clause gadget consists of a configuration of 3 point pairs surrounding variable gadgets corresponding to the variables in that clause (see Figure 4). The placement of the 3 point pairs depends on whether the literals appear positively or negatively.

Consider clause $c_i$ which consists of variables $x_j$ and $x_k$. We create a pair of points $\{a_i, b_i\}$, each of which will be placed next to variable $x_j$. If $x_j$ appears negatively in $c_i$, then
we place $a_i$ at distance 1 away from an endpoint of a green edge of $x_j$ and place $b_i$ at distance 1 away from the other endpoint of the green edge (see Figure 4c). If $x_j$ appears positively in $c_i$, then we do the same thing at the endpoints of a red edge (see Figure 4a). Then, we create a pair of points $\{d_i, e_i\}$ and follow the same procedure for variable $x_k$. Finally, for each clause gadget we create a pair of points $\{f_i, g_i\}$ and place them at a distance of 1 from certain variable gadget points chosen based on how many literals appear positively in clause $c_i$: zero, one, or two. The placement of $\{f_i, g_i\}$ for all three cases can be found in Figure 4.

As a technical note, to complete the construction, all clause gadget points placed around the same variable gadget vertex are separated from each other by distance 2, and these points are at distance only 1 from the nearest variable gadget vertex. This will ensure that the optimal solution will not link any two clause gadget points to each other. Also, note that we use the metric completion to define the weights of the rest of the edges in the graph.

Connecting all points except those associated with clause gadgets into two MSTs has a base cost of $(2n - 2)L + 4n + 2\binom{m}{2}M$. Now note that a clause evaluates to true if and only if it costs 7 units to attach the clause gadget points to the backbones. A clause evaluates to false if and only if it costs 9 units to attach the clause gadget points to the backbones (see Figure 4). Thus, it is now apparent that there exists a truth assignment in 2SAT with $k$ clauses satisfied if and only if there exists a solution to the Min-Sum 2-MST problem with cost $(2n - 2)L + 4n + 2\binom{m}{2}M + 7k + 9(m - k)$.

An $O(1)$-approximation algorithm for Min-Sum 2-MST problem.
Compute $MST(S)$, a minimum spanning tree on all $2n$ points. Imagine removing the heaviest edge, $h$, from $MST(S)$. This leaves us with two trees; $T_1$ and $T_2$. Perform a preorder traversal on $T_1$, assigning nodes to $S_1$ as long as there is no conflict. If there is a conflict ($q_i$ is reached in the traversal and $p_i$ was already assigned to $S_1$) then assign the node to $S_2$. Repeat this for $T_2$. We then return $S_1$ and $S_2$ as our approximate partition. In order to help analyze this algorithm, we will think of the nodes assigned to $S_1$ as colored red (set $R$) and the nodes assigned to $S_2$ as colored blue (set $B$).

Case 1: All nodes in $T_1$ are of the same color and all nodes in $T_2$ are of the same color.
\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{figure3}
\caption{Truth assignment.}
\end{figure}

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{figure4}
\caption{Placement of clause gadget points and extra cost incurred to incorporate clause gadget points into two MSTs once a truth assignment over the variables is fixed.}
\end{figure}

This partition is optimal. To see this, note that the weight of $\text{MST}(S) \setminus \{h\}$ is a lower bound on the cost of the optimal solution as it is the cheapest way to create two trees, the union of which span all of the input nodes. Since each tree is single colored, we know that each tree must have $n$ points, exactly one from each pair, and thus is also feasible to our problem.

\textbf{Case 2:} One tree is multicolored and the other is not.

Let $OPT$ be the optimal solution. Suppose WLOG that $T_1$ contains only red nodes and $T_2$ contains both blue and red nodes. Then, $\exists i : p_i, q_i \in T_2$. Imagine also constructing an MST on each side of the optimal partition. By definition, in the MSTs built over the optimal partition, at least one point in $T_2$ must be connected to a point in $T_1$. This implies that the weight of the optimal solution is at least as large as $|h|$, as $h$ is the cheapest edge which spans the cut $(T_1, T_2)$. Therefore, $|h| \leq |OPT|$.

Consider $\text{MST}(R)$. By the Steiner property, we have that an MST over a subset $U \subseteq S$ has weight at most $\alpha[\text{MST}(S)]$ where $\alpha$ is the Steiner ratio of the metric space. Recall that $|\text{MST}(S) \setminus \{h\}| \leq |OPT|$. In this case, since $|h| \leq |OPT|$, we have that $|\text{MST}(R)| \leq \alpha[\text{MST}(S)] \leq 2\alpha|OPT|$.

Next, consider building $\text{MST}(B)$. Since no blue node exists in $T_1$, there does not exist
In this section the objective is to

2.2 Min-max

I

Using the above case analysis, we have the following theorem.

Theorem 6. There exists a 3-approximation algorithm for the Bottleneck 2-MST problem.

Remark: In a general metric space $\alpha = 2$ and in the Euclidean plane $\alpha \leq 1.3546$ [11].

2.2 Min-max

In this section the objective is to \(\min - \max\{|MST(S_1)|, |MST(S_2)|\}\).

Theorem 3. The Min-Max 2-MST problem is weakly NP-Hard in the Euclidean plane.

[The proof is in the appendix.]

Theorem 4. There exists a 4$\alpha$-approximation for the Min-Max 2-MST problem. [The proof is in the appendix.]

2.3 Bottleneck

In this section the objective is to \(\min - \max\{|\lambda(S_1)|, |\lambda(S_2)|\}\).

Lemma 5. Given \(n\) pairs of points on a line where consecutive points on the line are unit separated, there exists a partition of these points into two sets, \(S_1\) and \(S_2\), such that \(\max\{|\lambda(S_1)|, |\lambda(S_2)|\} \leq 3\).

Proof. The proof will be constructive, using Algorithm 1. We partition the points into \(n\) disjoint buckets, each bucket containing two consecutive points.

Observe that at the end of Algorithm 1, each bucket has exactly one red point and one blue point. Thus, the maximum distance between any two points of the same color is 3. ◀

Theorem 6. There exists a 3-approximation algorithm for the Bottleneck 2-MST problem on a line.

Proof. Note that if the leftmost \(n\) points do not contain two points from the same pair, then it is optimal to let \(S_1\) be the leftmost \(n\) points and \(S_2\) be the rightmost \(n\) points. Suppose now that the leftmost \(n\) points contain two points from the same pair. We run Algorithm 1 on the input. Imagine building two bottleneck spanning trees over the approximate partition (coloring) as well as over the optimal partition. Let \(\lambda\) be the longest edge (between two points of the same color) in our solution and \(\lambda^*\) be the longest edge in the optimal solution.

Consider any two consecutive input points \(s_i\) and \(s_{i+1}\) on the line. We first show that \(|\lambda^*| \geq |s_is_{i+1}|\) by arguing that the optimal solution must have an edge that covers the
Color the leftmost point, p, red
Let p' be the point that is in p's bucket
Let R be a set of red points and B be a set of blue points
R ← {p}; B ← ∅
while There exists an uncolored point do
  while p' is uncolored do
    if p is red then
      Color p's pair, q, blue
      B ← B ∪ {q}
      p ← q
    else
      Let p'' be the point in p's bucket
      Color p'' red
      R ← R ∪ {p''}
      p ← p''
    end
  end
  Find the leftmost uncolored point x and color it red. Let x' be the point in x's bucket
  p ← x; p' ← x'
end
S₁ ← R; S₂ ← B
return \{S₁, S₂\}

Algorithm 1: Coloring points on a line.

interval \([s_i, s_{i+1}]\). Suppose to the contrary that no such edge exists. This means that \(s_i\) is connected to \(n-1\) points only to its left and \(s_{i+1}\) is connected to \(n-1\) points only to its right. This contradicts the assumption that the leftmost \(n\) points contain two points from the same pair.

Let the longest edge in our solution be defined by two points, \(p_i\) and \(p_j\). Consider the number of input points in interval \([p_i, p_j]\). Input points in this interval other than \(p_i\) and \(p_j\) will have a different color than \(p_i\) and \(p_j\). It is easy to see that if \([p_i, p_j]\) consists of two input points, that \(|\lambda^*| = |\lambda|\), and if \([p_i, p_j]\) consists of three input points, that \(|\lambda^*| \geq |\lambda|/2\]. We know by lemma 5 that \([p_i, p_j]\) can consist of no more than four input points. In this last case, \(|\lambda^*|\) must be at least the length of the longest edge of the three edges in \([p_i, p_j]\). Thus, we see that \(|\lambda^*| \geq |\lambda|/3\).

\[\text{Theorem 7. There exists a 9-approximation algorithm for the Bottleneck 2-MST problem in a metric space.}\]

\[\text{Proof.}\] First, we compute \(MST(S)\) and consider the heaviest edge, \(h\). The removal of this edge induces a cut that separates the nodes into two sets, \(H_1\) and \(H_2\). If \(\exists i : p_i, q_i \in H_j\) for \(1 \leq i \leq n\) and \(1 \leq j \leq 2\), then we let \(S_1 = H_1\) and \(S_2 = H_2\) and return \(S_1\) and \(S_2\). Let \(\lambda^*\) be the heaviest edge in the bottleneck spanning trees built on the optimal partition. Note that \(MST(S)\) lexicographically minimizes the weight of the \(k\)th heaviest edge, \(1 \leq k \leq 2n - 1\), among all spanning trees over \(S\), and thus the weight of the heaviest edge in \(MST(S) \setminus \{h\}\) is a lower bound on \(|\lambda^*|\). Thus, in this case, our solution is clearly feasible and is also optimal as \(MST(S_1)\) and \(MST(S_2)\) are subsets of \(MST(S) \setminus \{h\}\).

Now suppose \(\exists j \in \{1, 2\} : p_i, q_i \in H_j, 1 \leq i \leq n\). This means that \(|\lambda^*| \geq |h|\). In
this case, we compute a bottleneck TSP tour on the entire point set. It is known that a bottleneck TSP tour with bottleneck edge $\lambda$ can be computed from $\text{MST}(S)$ so that $|\lambda| \leq 3|h| \leq 3|\lambda^*|$. 

Next we run Algorithm 1 on the TSP tour and return two paths, each having the property that the largest edge has weight no larger than $9|\lambda^*|$. \hfill

**Remark:** Consider the problem of partitioning the point set into two sets, $S_1$ and $S_2$, such that $p_i \in S_1$ if and only if $q_i \in S_2$, and computing a bottleneck TSP tour on $S_1$ and on $S_2$. The objective is to minimize the heaviest edge (across both paths). Let the heaviest edge in the bottleneck TSP tours built on the optimal partition be $\lambda^*$. The above algorithm gives a 9-approximation to this problem as well because the algorithm returns two Hamilton paths and we know that (using the notation in the above proof) $|\lambda^*| \leq |\lambda^*|$. Thus, $|\lambda| \leq 9|\lambda^*| \leq 9|\lambda^*|$. 

The following is a generalization of Lemma 5. Let $S = \{S_1, S_2, \ldots, S_n\}$ be a set of $n$ $k$-tuples of points on a line and let $S$ be the point set. Each set $S_i$, $1 \leq i \leq n$, must be colored with $k$ colors. That is, no two points in set $S_i$ can be of the same color. Consider two consecutive points of the same color, $p$ and $q$. We show that there exists a polynomial time algorithm that colors the points in $S$ so that the number of input points in interval $(p, q)$ is $O(k)$.

**Lemma 8.** There exists a polynomial time algorithm to color $S$ so that for any two consecutive input points of the same color, $p$ and $q$, the interval $(p, q)$ contains at most $2k - 2$ input points. [The proof is in the appendix.]

### 3 Matchings

Given a set $S$ of $n$ pairs of points in a metric space, partition the point set $S$ into two sets, $S_1$ and $S_2$, such that for each pair $\{p_i, q_i\}$, $p_i \in S_1$ if and only if $q_i \in S_2$. Let $M(X)$ be the minimum weight matching on point set $X$ and $|M(X)|$ be the cost of the matching. Let $\lambda(X)$ be the bottleneck edge (longest edge) in a matching on point set $X$ and $|\lambda(X)|$ be the cost of the bottleneck edge.

#### 3.1 Minimum Sum

The objective is to minimize $|M(S_1)| + |M(S_2)|$.

**Theorem 9.** There exists a 2-approximation for the Min-Sum 2-Matching problem in general metric spaces.

**Proof.** First, note that the weight of the minimum weight perfect matching on $S$, $M^*$, which excludes edges $(p_i, q_i) \forall i$ is a lower bound on $|\text{OPT}|$. Consider a minimum weight one of a pair matching where we think of each pair as a single node and have the weight of an edge between two nodes be the smallest weight edge of the four edges that exist between the two pairs. The weight of the minimum weight one of a pair matching, $\tilde{M}$, is a lower bound on the weight of the smaller of the matchings of OPT and therefore has weight at most $|\text{OPT}|/2$.

Our algorithm computes $\tilde{M}$, and sets the points involved in this matching to be in $S_1$ and sets $S_2 = S \setminus S_1$. We return the partition $S_1, S_2$ as our approximate solution. We have that $|M(S_1)| = |\tilde{M}| \leq |\text{OPT}|/2$. To bound $|M(S_2)|$, consider the multigraph $G = (V = S, E = M^* \cup \tilde{M})$. All $v \in S_2$ have degree 1 (from $M^*$), and all $u \in S_1$ have
Figure 5  \( \frac{\text{APX}}{\text{OPT}} \approx 2 \)

degree 2 (from \( M^* \) and \( \hat{M} \)). For each \( v_i \in S_2 \), either \( v_i \) is matched to \( v_j \in S_2 \) by \( M^* \), or \( v_i \) is matched to \( u_i \in S_1 \) by \( M^* \). In the former case we can consider \( v_i \) and \( v_j \) matched in \( S_2 \) and charge the weight of this edge to \( |M^*| \). In the latter case, note that each \( u_i \in S_1 \) is part of a unique cycle, or a unique path. If \( u_i \in S_1 \) is part of a cycle then no vertex in that cycle belongs to \( S_2 \) due to the degree constraint. Thus, if \( v_i \in S_2 \) is matched to \( u_i \in S_1 \), \( u_i \) is part of a unique path whose other terminal vertex \( x \) belongs to \( S_2 \), due to the degree constraint. We can consider \( v_i \), and \( x \) matched and charge the weight of this edge to the unique path connecting \( v_i \) and \( x \) in \( G \). Thus, \( |M(S_2)| \) can be charged to \( |M^* \cup \hat{M}| \) and has weight at most 1.5\( |OPT| \).

Therefore, our partition guarantees \( |M(S_1)| + |M(S_2)| \leq 2|OPT| \). Figure 5 shows the approximation factor using our algorithm is tight.

3.2 Min-max

In this section the objective is to \( \min - \max \{|M(S_1)|, |M(S_2)|\} \).

Theorem 10. The Min-Max 2-Matching problem is weakly NP-Hard in the Euclidean plane. [The proof is in the appendix.]

Theorem 11. The approximation algorithm for the Min-Sum 2-Matching problem serves as a 3-approximation for the Min-Max 2-Matching problem in general metric spaces. [The proof is in the appendix.]

3.3 Bottleneck

In this section the objective is to \( \min - \max \{|\lambda(S_1)|, |\lambda(S_2)|\} \).

Theorem 12. There exists a 3-approximation to the Bottleneck 2-Matching problem in general metric spaces. [The proof is in the appendix in section B.]

4 TSP Tours

Given a set \( S \) of \( n \) pairs of points in a metric space, partition the point set \( S \) into two sets, \( S_1 \) and \( S_2 \), such that for each pair \( \{p_i, q_i\} \), \( p_i \in S_1 \) if and only if \( q_i \in S_2 \). Let \( TSP(X) \) be a TSP tour on point set \( X \) and \( |TSP(X)| \) be the cost of the tour. Let \( \lambda(X) \) be the bottleneck edge in a TSP tour on point set \( X \) and \( |\lambda(X)| \) be the cost of the bottleneck edge.

In this section we consider partitioning a set of input pairs of points so that the cost function of a pair of TSP tours, one built on each set of the partition, is minimized. It is interesting to note the complexity difference emerging here. In prior sections, the structures to be computed on each set of the partition were computable exactly in polynomial time. Thus, the decision versions of these problems, which ask if there exists a partition such that some cost function over the pair of structures is at most \( k \), are easily seen to be in \( \text{NP} \). However, when the cost function is over a set of TSP tours or bottleneck TSP tours, this is no longer the case. That is, suppose that a non-deterministic Touring machine could in polynomial time, for a point set \( S \) and \( k \in \mathbb{R} \), return a partition for which it claimed the
cost of the TSP tours generated over both sets is at most $k$. Unless $P = NP$, the verifier cannot in polynomial time confirm that this is a valid solution, and therefore the problem is not in NP. Thus, the problems considered in this section are all NP-Hard.

### 4.1 Minimum Sum

In this section the objective is to minimize $|TSP(S_1)| + |TSP(S_2)|$.

Let a $\beta$-factor approximate TSP tour on set $X$ be denoted $\overline{TSP}(X)$.

1. Compute $\overline{TSP}(S)$.
2. Let $2k$ be the maximum even number not exceeding $(2 + \frac{1}{\beta}) \beta$. Enumerate all ways of decomposing $\overline{TSP}(S)$ into $2k$ connected components: for each decomposition, assign the nodes from the components to $S_1$ and $S_2$ alternately (i.e. assign all nodes in component one to $S_1$, all nodes in component two to $S_2$, etc.). If this partition is infeasible, then skip to the next decomposition; otherwise compute $\overline{TSP}(S_1)$ and $\overline{TSP}(S_2)$.
3. Compute a random feasible partition, $S = \hat{S}_1 \cup \hat{S}_2$, and compute $\overline{TSP}(\hat{S}_1)$ and $\overline{TSP}(\hat{S}_2)$.
4. Among all pairs of tours produced in steps 2 and 3, choose the pair of minimum sum.

**Algorithm 2:** Algorithm $A(\mu, \beta)$. $0 < \mu < 1$ and $\beta > 1$.

We will show for $\beta > 1$ and for the proper choice of $\mu$, that Algorithm 2 gives a $3\beta$-approximation for the Min-Sum 2-TSP problem. Fix a constant $\mu < 1$. Let $OPT$ be the optimal partition $S = S_1 \cup S_2$. Let $d(S_1, S_2)$ be the minimum point-wise distance between sets $S_1$ and $S_2$. We call an instance of the problem $\mu$-separable if there exists a feasible partition $S = S_1 \cup S_2 : d(S_1, S_2) \geq \mu(|TSP(S_1)| + |TSP(S_2)|)$.

Let $APX$ be the partition returned by our approximation. We will show that if $S$ is not $\mu$-separable, then $|APX| \leq \frac{2}{1+\mu} |OPT|$ (see Lemma 13) and that if $S$ is $\mu$-separable, then $|APX| \leq \frac{\beta}{4^\mu} |OPT|$ (see Lemma 14). Supposing both of these are true, then the approximation factor of our algorithm is $max\{\frac{\beta}{4^\mu}, \frac{128}{1+\mu}\}$. One can easily verify that $\mu = 1/12$ is the minimizer which gives the desired $3\beta$ factor. The following lemma states that if $S$ is not $\mu$-separable, then any feasible partition yields a “good” approximation.

**Lemma 13.** If $S$ is not $\mu$-separable, then $|APX| \leq \frac{2}{1+\mu} |OPT|$.

**Proof.** If $S$ is not $\mu$-separable, then for any feasible partition $S = S_1 \cup S_2$ we have $d(S_1, S_2) \leq \mu(|TSP(S_1)| + |TSP(S_2)|)$. In particular, for the partition induced by the optimal solution, $S = S_1^* \cup S_2^*$, $d(S_1^*, S_2^*) \leq \mu(|TSP(S_1^*)| + |TSP(S_2^*)|)$. Then,

$$|TSP(S)| \leq |OPT| + 2d(S_1^*, S_2^*) \leq |OPT| + 2\mu(|TSP(S_1^*)| + |TSP(S_2^*)|) \leq |OPT| + 4\mu|TSP(S)|.$$

Hence, when $\mu < \frac{1}{3}$, $|TSP(S)| \leq \frac{4}{1+\mu} |OPT|$. Let $S = \hat{S}_1 \cup \hat{S}_2$ be the random feasible partition computed by $A(\mu, \beta)$. Then, as we are returning the best partition between $\hat{S}_1 \cup \hat{S}_2$ and all $O(n^{2k})$ partitions of $\overline{TSP}(S)$, we have $|APX| \leq \beta(|TSP(\hat{S}_1)| + |TSP(\hat{S}_2)|) \leq 2\beta|TSP(S)| \leq \frac{2\beta}{1+\mu} |OPT|$.

The following lemma states that if $S$ is $\mu$-separable, then any witness partition to the $\mu$-separability of $S$ gives a “good” approximation.

**Lemma 14.** If $S$ is $\mu$-separable, then $|APX| \leq \frac{\beta}{4^\mu} |OPT|$.

**Proof.** Suppose we successfully guessed a partition $X_0 = S_1^0 \cup S_2^0$ that is a “witness” to the $\mu$-separability of $S$ (we will show how to guess $X_0$ later).
Case 1: $\text{OPT} = X_0$. Then $|\text{APX}| \leq \beta(|TSP(S_0)| + |TSP(S_2^0)|) = \beta(|TSP(S_1^0)| + |TSP(S_2^0)|) = \beta|\text{OPT}|$.

Case 2: $\text{OPT} \neq X_0$. Then for $i = 1, 2$, $S_i^* \neq S_0^0$, which means each tour in $\text{OPT}$ must contain at least 2 edges crossing the cut $(S_1^0, S_2^0)$, hence the optimal solution must contain at least 4 edges crossing the cut $(S_1^0, S_2^0)$. So $|\text{OPT}| \geq 4d(S_1^0, S_2^0) \geq 4\mu(|TSP(S_1^0)| + |TSP(S_2^0)|) \geq \frac{4\mu}{\beta}|\text{APX}|$. Equivalently, $|\text{APX}| \leq \frac{\beta}{4\mu}|\text{OPT}|$.

The next two lemmas show how to guess a witness partition $X_0$ in polynomial time. First, we show that if $S$ is $\mu$-separable with a witness partition $X_0$, then $TSP(S)$ cannot cross this partition “too many” times.

**Lemma 15.** Let $TSP(S)$ be an $\beta$-factor approximation for $TSP(S)$. Also, suppose $S$ is $\mu$-separable with witness $X_0$. Then $TSP(S)$ crosses the cut $(S_1^0, S_2^0)$ at most $(2 + \frac{1}{\mu})\beta$ times.

**Proof.** One can construct a TSP tour for $S$ by adding two bridges to $TSP(S_1)$ and $TSP(S_2)$, thus we have $|TSP(S)| \leq |TSP(S_1)| + |TSP(S_2)| + 2d(S_1^0, S_2^0) \leq (2 + \frac{1}{\mu})d(S_1^0, S_2^0)$. Also, suppose $TSP(S)$ crosses the cut $(S_1^0, S_2^0)$ $2k$ times. Then, $2kd(S_1^0, S_2^0) \leq |TSP(S)| \leq \beta|TSP(S)|$. Combining the above two inequalities, we obtain $2k \leq (2 + \frac{1}{\mu})\beta$.

The next lemma completes our proof.

**Lemma 16.** Suppose $S$ is $\mu$-separable. Let $X_0$ be any partition which serves as a “witness”. Then, in step 2 of $A(\mu, \beta)$, we will encounter $X_0$ at some stage.

**Proof.** Given a nonnegative integer $k$ and a TSP tour $P$, define $\Pi(P, k) = \{X: X$ is a feasible partition and $P$ crosses $X$ at most $k$ times$\}$. By Lemma 15, we know $X_0 \in \Pi(TSP(S), (2 + \frac{1}{\mu})\beta)$. Since step 2 of $A(\mu, \beta)$ is actually enumerating all partitions in $\Pi(TSP(S), (2 + \frac{1}{\mu})\beta)$, we are done.

Note that step 2 considers $O(n^{2k}) = O(n^{14\beta})$ decompositions and for each partition that is feasible, we compute two approximate TSP tours. Suppose the running time to compute a $\beta$-factor TSP tour on $n$ points is $h_\beta(n)$. Then the worst case running time of Algorithm 2 is $O(h_\beta(2n)n^{14\beta})$. Thus, we have the following Theorem.

**Theorem 17.** For any $\beta > 1$, the algorithm $A(\frac{1}{\beta}, \beta)$ is a $3\beta$-approximation for the Min-Sum 2-TSP problem with running time $O(h_\beta(2n)n^{14\beta})$.

**Remark:** If $S$ is in the Euclidean plane then $\beta = 1 + \epsilon$ for some $\epsilon > 0$ [13] yielding a $(3 + \epsilon)$-approximation and if $S$ is in a general metric space then $\beta = 3/2$ [6] yielding a 4.5-approximation. In both cases $h_\beta(2n)$ is polynomial.

### 4.2 Min-Max

In this section the objective is to $\text{min} - \text{max}\{|TSP(S_1)|, |TSP(S_2)|\}$.

**Theorem 18.** There exists a $6\beta$-approximation to the Min-Max 2-TSP problem, where $\beta$ is the approximation factor for TSP in a certain metric space. [The proof is in the appendix.]

### 4.3 Bottleneck

In this section the objective is to $\text{min} - \text{max}\{|\lambda(S_1)|, |\lambda(S_2)|\}$.

**Theorem 19.** There exists an $18$-approximation algorithm for the Bottleneck 2-TSP problem. [The proof is in the appendix.]
References


\section*{A \quad Additional Proofs}

**Theorem 3.** The Min-Max 2-MST problem is weakly NP-Hard in the Euclidean plane.

**Proof.** The reduction is from Partition. In the partition problem we are given a set of \( n \) integers, \( P = \{x_1, x_2, \ldots, x_n\} \), with the objective of deciding if there exists a partition \( P = P_1 \cup P_2 \), with \( \sum_{i \in P_1} x_i = \sum_{j \in P_2} x_j \). Let \( M = \sum x_i \) be the sum of all integers in a given instance of partition. Then, we can construct an instance of Min-Max 2-MST in the plane with 2\( n \) pairs of points as follows. Place \( n \) point pairs \( \{p_i, q_i\}_{i=1}^n \) along two \( \epsilon \)-separated horizontal lines, such that \( p_i, q_i \) are vertically adjacent, with horizontal separation of \( M \) between consecutive pairs. Then, for each \( x_i \) in the instance of Partition we place a point \( p_{n+i} \) at distance \( x_i \) from \( p_i \) and its corresponding pair \( q_{n+i} \) at distance \( \epsilon/2 \) from \( q_i \), and \( p_i \) (See Figure 6).

Imagine building an MST on each side of the optimal partition for this instance. Notice that for pairs \( \{p_i, q_i\}_{i=1}^n \), edges \( (p_i, p_{i+1}) \) and \( (q_i, q_{i+1}) \) will exist in the optimal solution for \( 1 \leq i \leq n-1 \) and serve as “backbones” of the two MSTs. For each remaining point pair \( i \), \( n+1 \leq i \leq 2n \), as \( \{p_i, q_i\} \) can not be assigned to the same tree, by definition, one tree will incur an “extra cost” of \( \epsilon/2 \) and the other will incur an “extra cost” of \( x_i \). Therefore, any algorithm which minimizes the maximum weight of either tree also minimizes \( \max\{\sum_{i \in P_1} x_i, \sum_{j \in P_2} x_j\} \) across all partitions \( P_1 \cup P_2 \).

Thus, by choosing \( \epsilon \) small enough, we can guarantee that any proposed exact algorithm to Min-Max 2-MST will produce two trees, each of weight \( (n-1)M + M/2 + o(1) \), if and only if there exists a feasible solution to the partition instance.

![Figure 6 Set up of Min-Max 2-MST instance given an instance of Partition: \( \{x_1, x_2, \ldots, x_n\} \).](image)

\section{Additional Proofs}

**Theorem 4.** There exists a \( 4\alpha \)-approximation for the Min-Max 2-MST problem.

**Proof.** We use the same algorithm as we did for the Min-Sum 2-MST problem. The approximation factor is dominated by case 2 in the Min-Sum 2-MST analysis. For the Min-Max objective function, we have that \( \max\{|\text{MST}(B)|, |\text{MST}(R)|\} \leq \alpha |\text{MST}(S)| \) and that \( |\text{MST}(S)| \leq 4|\text{OPT}| \). Thus, \( \max\{|\text{MST}(B)|, |\text{MST}(R)|\} \leq 4\alpha |\text{OPT}| \).

**Lemma 8.** There exists a polynomial time algorithm to color \( S \) so that for any two consecutive input points of the same color, \( p \) and \( q \), the interval \( (p, q) \) contains at most \( 2k - 2 \) input points.

Proof. The reduction is from PARTITION: given a set $P = \{x_1, x_2, ..., x_n\}$ of $n$ integers, decide if there exists a partition $P = P_1 \cup P_2$, with $\sum_{i \in P_1} x_i = \sum_{j \in P_2} x_j$. Let $M = \sum_i x_i$. Given any instance $P$ of PARTITION, we create a geometric instance of the Min-Max 2-Matching problem, as shown in Figure 7.

We place $n$ point pairs $(p_i, q_i)$ along two $\epsilon$-separated horizontal lines, such that $p_i, q_i$ are vertically adjacent, with horizontal separation of $M$ between consecutive pairs. Then, for each $x_i$ in the instance of PARTITION we place a point $p_{n+i}$ at distance $x_i$ from $p_i$, and its corresponding pair $q_{n+i}$ at distance $\epsilon/2$ from both $q_i$ and $p_i$.

Notice that any solution which minimizes the weight of the larger matching created only uses edges between points of the same "cluster" $\{p_i, q_i, p_{n+i}, q_{n+i}\}$ for $i$. Any edge between two clusters $\{p_i, q_i, p_{n+i}, q_{n+i}\}, \{p_j, q_j, p_{n+j}, q_{n+j}\}$, $i \neq j$ costs at least $M$ and if matching edges are chosen within clusters the entire matching can be constructed with cost at most $M + o(1)$ for $\epsilon > 0$ chosen small enough.

Within each cluster an assignment will have to be made, that is, WLOG, $\{p_i, p_{n+i}\} \in S_1, \{q_i, q_{n+i}\} \in S_2$ or $\{p_i, q_{n+i}\} \in S_1, \{q_i, p_{n+i}\} \in S_2$. Therefore, any algorithm that minimizes the maximum weight of either matching also minimizes $\max(\sum_{i \in P_1} x_i, \sum_{j \in P_2} x_j)$ across all partitions $P_1 \cup P_2$. Thus, for $\epsilon > 0$ chosen small enough the instance of partition is solvable if and only if the weight of the larger matching created is at most $\frac{M}{2} + o(1)$.


Proof. In this case we are concerned only with the larger of the two matchings returned by our approximation, $M(S_2)$, which, as described above, has weight bounded above by $|\hat{M}| + |M^*|$. However, in this case, under the new cost function $|\hat{M}| \leq |OPT|$, and $|M^*| \leq 2|OPT|$. Therefore, $|M(S_2)| \leq |\hat{M}| + |M^*| \leq 3|OPT|$. The example illustrated in Figure 5 shows the approximation factor achieved by this algorithm is tight.

Theorem 12. There exists a $3$-approximation to the Bottleneck 2-Matching Problem in general metric spaces. [See section B of the appendix.]

Theorem 18. There exists a $6\beta$-approximation to the Min-Max 2-TSP problem, where $\beta$ is the approximation factor for TSP in a certain metric space.
Figure 7 Set up of the Min-Max 2-Matching instance given an instance of \textit{Partition}: \{x_1, x_2, \ldots, x_n\}.

Proof. We use the same algorithm, and return the same partition, \(S_1 \cup S_2\), as in Section 4.1. Let \(APX\) be the partition returned, and \(|APX|\) be the cost of the larger TSP on both sets of \(APX\). Let \(OPT\) be the optimal solution. Note that \(|APX| \leq 3\beta(|TSP(S_1)| + |TSP(S_2)|) \leq 6\beta|OPT|\) as \(|TSP(X_1)| + |TSP(X_2)| \leq 2\max\{TSP(X_1), TSP(X_2)\}\) for any \(X = X_1 \cup X_2\).


Proof. We remarked in section 2.3 that there exists a 9-approximation to the problem of finding a partition that minimizes the weight of the bottleneck edge on two Hamilton paths built on the partition. A Hamilton path can be converted into a Hamilton cycle by at most doubling the weight of the bottleneck edge in the Hamilton path. This yields an 18-approximation to the Bottleneck 2-TSP problem.

Bottleneck 2-Matching

We begin with a lemma concerning the structure of a feasible solution. Let \(S = S_1 \cup S_2\) be any feasible partition to the Bottleneck 2-Matching instance. Let \(M_B(X)\) be a minimum bottleneck matching on point set \(X\). Construct a graph \(G = (V, E)\) where \(V = S\) and \(E = (M_B(S_1) \cup M_B(S_2) \cup \{p_i, q_i\}_{i=1}^n)\) is the union of any pair of optimal bottleneck matchings on \(S_1\) and \(S_2\) and the edges \((p_i, q_i)\) for all \(i\).

\textbf{Lemma 20}. \(G\) is a 2-factor such that each input pair is contained in exactly one cycle, and each cycle contains an even number of input pairs.

Proof. The edge set of \(G\) is the union of two disjoint perfect matchings over \(S\). Therefore, each node has degree exactly 2 and \(G\) is by definition a 2-factor. Also, by definition of a 2-factor, each input point \(p_i\) is part of a unique cycle, and in this case, as each node \(p_i\) has an edge of the form \((p_i, q_i)\) incident to it, therefore, the point \(q_i\) must be contained in the same cycle as \(p_i\) for all \(i\). Thus, each input pair is contained in the same unique cycle in \(G\).

To see that each cycle contains an even number of input pairs, imagine coloring the nodes of \(S_1\) red and the nodes of \(S_2\) blue. Suppose by contradiction that some cycle
Thus, any edge not of the form $(p_i, q_i)$ and 2-coloring the edges of $M_B(S_1) \cup M_B(S_2)$, as the edges of this cycle alternate between the form $(p_i, q_i)$ and edges in $M_B(S_1) \cup M_B(S_2)$. Since this contracted cycle has an odd number of edges, this 2-coloring is not possible, thus a contradiction.

Using the above structure lemma we will argue that we can compute a graph with the same properties and extract a feasible partition with constant factor approximation guarantees. Let $\tilde{M}_B(S)$ be the minimum weight (exactly) one of a pair bottleneck matching over $S$; note that edges of this matching go between points of $S$. Let $M_B(S)$ be the minimum weight bottleneck matching over $S$ (excluding the edges $(p_i, q_i) \forall i$). Let $\lambda$ (resp. $\hat{\lambda}$) be the heaviest edge used in $\tilde{M}_B(S)$ (resp. $M_B(S)$) and let $\lambda^*$ be the heaviest edge in a minimum weight bottleneck matching computed over each of the two sets in $OPT$. Note that $|\hat{\lambda}| \leq |\lambda^*|$ and $|\lambda| \leq |\lambda^*|$. Begin by constructing a graph $G = (V = S, E = M_B(S) \cup (p_i, q_i)_{i=1}^n)$, which is a 2-factor as its edge set is the union of two disjoint perfect matchings. Note, it will be the case that each input pair exists in the same unique cycle. If each cycle contains an even number of input pairs then this graph has the same structure as that described in Lemma 20 and thus we can extract a feasible partition from $G$. We will describe how to obtain this partition later. As $|\lambda^*| \geq |\lambda|$, this graph induces an optimal partition. On the other hand, if there exists a cycle with an odd number of input pairs (there must be an even number of such cycles) then we “merge” cycles of $G$ together into larger cycles until a point is reached in which each new “super-cycle” contains an even number of pairs. From this graph we can extract a constant factor approximation to an optimal partition.

**Lemma 21.** If $G$ contains at least one cycle with an odd number of input pairs, then it is possible to merge cycles of $G$ into super-cycles, each of which contains an even number of input pairs, such that the heaviest edge (excluding $(p_i, q_i) \forall i$) in any super-cycle has weight at most $3|\lambda^*|$.  

**Proof. (sketch)** Superimpose a subset of the edges in $\tilde{M}_B(S)$ over the nodes of $G$ in the following way. Consider only edges in $\tilde{M}_B(S)$ which have endpoints in different cycles. Treat each cycle in $G$ as a node and run Kruskal’s algorithm until all of the aforementioned edges of $\tilde{M}_B(S)$ are exhausted. This yields a forest on the cycles of $G$. It is easy to see that every cycle of $G$ containing an odd number of input pairs has an edge of $\tilde{M}_B(S)$ connecting it to some other cycle. This implies that it is possible to merge all cycles which are connected by an edge of $\tilde{M}_B(S)$ until one reaches a point where all cycles have an even number of pairs. We give a brief outline for this merging process.

Find any maximal path $P$ in $G$ which alternates edges of the form $M_B(S)$ and $\tilde{M}_B(S)$. Consider the cycles in $G$ containing the edges of $M_B(S)$ in $P$ (see Figure 8a). We will label the cycles in this path $C_1, C_2, \dotsc, C_k$ where $k$ is the number of cycles in the path. We will “stitch” the cycles together into a final super-cycle by making connections between pairs of cycles with odd subscripts in sorted order then by making connections between cycles with even subscripts in reverse sorted order. Now, remove all edges of $P$ and we are left with a super-cycle (see Figure 8b). Recall that the weight of each edge of $M_B(S)$ and $\tilde{M}_B(S)$ is a lower bound on $|\lambda^*|$. Consider the two nodes that define some edge $e$ created in the stitching process. Note that $e$ can be replaced by path of at most three edges from $M_B(S) \cup \tilde{M}_B(S)$. Thus, any edge not of the form $(p_i, q_i)$ in the super-cycle has weight at most $3|\lambda^*|$. It is
not difficult to see that the stitching process can be done while keeping the weight of the bottleneck edge at most $3|\lambda^*|$ regardless of whether $k$ is even or odd.

It is possible that many maximal paths share an edge with the same cycle $C_i$ in $G$ (see Figure 9a). These edges must all be different because, when only considering edges of $M_B(S) \cup \hat{M}_B(S)$, the degree of each node in $G$ is at most two. This implies that any edge created in one stitching process is not altered by another stitching process and thus stitching processes are independent of one another. All cycles associated with these paths will be merged into the same super-cycle (see Figure 9b), and since the merging processes are independent, the bottleneck edge created is still of weight no larger than $3|\lambda^*|$. \hfill $\blacksquare$

Now that each cycle contains an even number of input pairs, all that is left to show is how to create a feasible partition from these cycles so that the weight of the heaviest edge in the bottleneck matching computed on either side of the partition is at most $3|\lambda^*|$. Notice that for each cycle, every other edge is of the form $(p_i, q_i)$. We will 2-color the nodes of each cycle red and blue so that if two nodes share an edge of the form $(p_i, q_i)$, they must be of different color, and if they share an edge not of the form $(p_i, q_i)$ they must be of the same color. Assign the red nodes to $S_1$ and the blue nodes to $S_2$. This partition is clearly feasible and the weight of the heaviest edge in the matchings created on either side is equal to the weight of the heaviest edge not of the form $(p_i, q_i)$ among all of the cycles we have created; this edge has weight at most $3|\lambda^*|$. \hfill $\blacksquare$

**Theorem 12.** There exists a 3-approximation to the Bottleneck 2-Matching problem in general metric spaces.