Tight Analysis of Local Search Heuristics for the Red-Blue Median Problem

by

Yifeng Zhang

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computing Science

University of Alberta

Abstract

Budgeted Red-Blue Median is a generalization of classic k-Median in that there are two sets of facilities, say $\mathcal{R}$ and $\mathcal{B}$, that can be used to serve clients located in some metric space. The goal is to open k_r facilities in $\mathcal{R}$ and k_b facilities in $\mathcal{B}$ for some given bounds k_r, k_b and connect each client to their nearest open facility in a way that minimizes the total connection cost.

We extended the work by Hajiaghayi, Khandekar, and Kortsarz [2012] and show that a local search technique can be used to obtain a $(5+\epsilon)$-approximation for any $\epsilon > 0$. This is an improvement over their analysis and beats the previous best approximation guarantee of 8 by Swamy [2014].

We also present a matching lower bound showing that for every $p \geq 1$, there are instances of Budgeted Red-Blue Median with local optimum solutions for the p-swap heuristic whose cost is $5 + \Omega\left(\frac{1}{p}\right)$ times the optimum solution cost. Thus, our analysis is essentially tight. In addition, the lower bound result extends to allowing up to a constant t number of colors in facilities. We show that the natural local search algorithm has a lower bound of $2t + 1 + \Omega\left(\frac{1}{p}\right)$ in this case.

Acknowledgements

I thank lord God for my existence as well as the existence of this world. Life, is mysterious and wonderful, without it, none of this research would be possible.

I would also like to express my sincere gratitude to my advisor Prof. Zachary Friggstad. Words can not approximate the amount of gratitude I have towards his support and guidance that is essential for the completion of this Master study. I have learned so much about researching as well as being a human from him just by observing his patience, motivation and immense knowledge both in life and research.

Besides my advisor, I would like to thank the rest of my thesis committee: Prof. Mohammad R. Salavatipour, and Prof. Guohui Lin, for their insightful comments and encouragement, but also for the hard question which incented me to widen my research from various perspectives.

I thank my undergraduate supervisor Prof. Yangjun Chen for being an important influence in developing my passion for the field of computer science. It is with him that I conducted my first scientific research project.

I would like to thank the Department of Computing Science for providing me the opportunity of working as teaching assistant which is a valuable experience for my professional development.

Last but not the least, I like thank my parents for supporting me both spiritually and financially. Without them this is impossible.

Thanks the Communists party of People's Republic of China, for letting me go.

Contents

List of Figures

Chapter 1

Introduction

1.1 Problem Definition and Motivation

The problem of finding an efficient way of opening facilities to serve clients has received a lot of attention in recent research projects. While there are many similar versions of the problem, these problems typically involve opening facilities in a metric setting, following a certain set of rules where the goal is to minimize some specific cost function. Many of its variations are known to be NP-hard. Thus, often a feasible solution that was produced within a reasonable amount of time (relative to the input size) should be expected to be sub-optimal. An approximation algorithm for an optimization problem will produce a feasible solution within a reasonable time length (*e.g.* in polynomial time), and guarantees the cost of the solution is within some factor of the best possible value.

In a general setting of a facility location problem, one typical measure of a solution's quality is the distance between clients and the facility serving it. Furthermore, if we open many facilities then we can be near every client, so it is only natural to take into account of the number of facilities opened to measure the quality of a solution. Instead of setting a limit on the total number of facilities that could be opened, we sometimes associate with every facility a cost of opening that facility. These two measures are often referred to as service cost and facility cost. Combining various constraints on these two costs naturally results in variants of the general facility location problem. For example, in the classic k-MEDIAN problem, we are allowed to open a total of k facilities and the goal is to minimize the total distance between all clients to their nearest facility.

Our model of facility location is called Budgeted Red-Blue Median and is described as the following. We are given a metric space with a finite set of locations V and distances $d(u, v)$ between any $u, v \in V$. The distances are symmetric, non-negative, and satisfy triangle inequality: $d(u, v) + d(v, w) \leq d(u, w)$ for any $u, v, w \in V$. In the set of locations, we distinguish three types of location: the clients $\mathcal{C}$, the Red Facilities $\mathcal{R}$, and the Blue Facilities $\mathcal{B}$. We are also given budgets $k_r, k_b \in \mathbb{Z}$ for each type of facilities meaning that the number of facilities opened for each type cannot exceed k_r, k_b respectively. The goal is to open facilities to satisfy the budget constraint while minimizing the total service cost. The main difference here compared to the classic k-Median is that there are two types of facilities instead of one. Budgeted Red-Blue Median is a generalization of the k-Median because if we let all facilities have the same color it becomes same as the k-Median problem.

Facility location problems have a wide variety of applications and it is no exception for Budgeted Red-Blue Median. For example, a manufacturing company provides a component gear that can be made in two types of materials. Each material resource is only available at certain locations. The component is some what crucial and needs to be shipped to the clients but the company only has a limited budget for opening each type of facilities. Thus finding the best way to utilize these facilities by setting up shops near some of the material resources that can serve many clients while lower the cost of sending the products to the clients could potentially be very beneficial. This scenario corresponds to the problem we study in this thesis. Many similar models are heavily studied in other related works such as k-Median.

1.2 Notation and Preliminaries

In Budgeted Red-Blue Median, we define a feasible solution (R, B) such that $R \subseteq \mathcal{R}$ and $B \subseteq \mathcal{B}$ with $|R| = k_r, |B| = k_b$. The goal is to minimize:

$$cost(R \cup B) := \sum_{j \in \mathcal{C}} \min_{i \in R \cup B} d(i, j).$$

A *graph* G is an ordered pair (V, E) where V is a finite set of vertices (nodes) and E is a collection of unordered pairs of distinct vertices called edges. The two vertices u, v in an edge e are called the *endpoints* of e. Typically, each node v can be the endpoint of several edges, we call the number of such edges the degree of v. If the degree of v is 0, v is called an *isolated node*. A *weighted graph* is a graph $G = (V, E, w)$ where w is a weight function $w : E \to \mathbb{R}$. Given a subset $U \subseteq V$, the subgraph induced by U is the graph $G' = (U, E')$ where $E' = \{(u, v) \in E : u, v \in U\}$. A *walk* in a graph is a sequence of vertices $v_1, v_2, v_3, ..., v_k \in V$ such that $(v_i, v_{i+1}) \in E$ for all $0 < i < k$. A *path* is a walk where each vertex appear at most once in the walk. The *length* of a path is the total weight of all the edges in the path and for the unweighted graph, the length of a path is the total number of edges. We say a shortest path between u and v is a path starting at u and ending at v of minimum length.

A *partition* $\mathcal{P}$ of a set S is a collection of subsets of S where each part $A \in \mathcal{P}$ satisfies the following:

- $A \neq \emptyset$
- $A \cap B = \emptyset$ for any $B \in \mathcal{P} - \{A\}$
- $S = \cup_{A' \in \mathcal{P}} A'$

A partition $\mathcal{G}$ on a graph $G = (V, E)$ is a collection of induced subgraphs $G' = (V', E')$ of G where the node sets $\{V' : G' \subseteq G\}$ form a partition of V.

Given any metric space on a finite set of locations V and with distances $d(u, v)$ between locations $(u, v) \in V$, we can construct a weighted graph $G = (V, E)$ where there is a one-to-one correspondence between vertices and locations. An edge weight $d(u, v)$ is also assigned to each pair of vertices

$u, v \in V$. Similarly, given a graph $G = (V, E)$ with non-negative edge weights, the graph corresponds to a metric space where the distance $d(u, v)$ between the two location u, v is equal to the total edge weight of the shortest path between the two nodes $u, v \in V$ where the nodes corresponds to the locations.

An optimization problem is a problem in which each instance I has a set of feasible solutions F_I and objective function $M_I : F_I \to \mathbb{R}$ where the goal is to find some $f \in F_I$ that minimizes or maximizes $M_I(f)$. In this work, we only consider minimization problems and thus we will assume the goal is always to find a solution with the minimum cost. An $NP-$hard optimization problem has its decision version being $NP-$hard. That is, given an instance I and a value k, it is $NP-$hard to determine if there exists an $f \in F_I$ such that $M_I(f) \leq k$.

An $\alpha-$approximation algorithm with a value $\alpha \geq 1$ for an optimization problem is a polynomial time algorithm that produces a feasible solution f for any given instance I of the problem such that $M_I(f) \leq \alpha \min_{g \in F_I} M_I(g)$. We say an algorithm is a constant factor approximation algorithm if α is constant.

A *polynomial-time approximation scheme* ($PTAS$) is a type of approximation algorithm for optimization problems. A $PTAS$ is an algorithm which takes an instance of an optimization problem and a parameter $\epsilon > 0$ and, in polynomial time, produces a solution that is within a factor $1 + \epsilon$ of being optimal. The running time of a $PTAS$ is required to be polynomial in the input size for every fixed ϵ but can be different for different ϵ. For example, an $(1 + \epsilon)$-approximation algorithm running in time $O(n^{\frac{1}{\epsilon}})$ counts as a $PTAS$.

In this thesis, we discuss *local search* algorithms. While there is not a precise definition of local search algorithm, a typical local search algorithm starts with a feasible solution and iteratively improves the solution by searching through any solution lies within the range of the local space. We restrict the searching operation to run within polynomial time. A $p-$swap heuristic is referring to a specific tactical scheme that we use for the local search algorithm in this thesis and many other local search algorithms on other similar problems. With a given parameter p, a local search determines its searching range and therefore affects its running time as well as the approximation guarantee. Such setting has the ability to affect quality of the result solution, for example, if p is set to be large enough, the local search algorithm may guarantee to output the optimum solution, but in the mean time it may become very slow such that exceeds polynomial time.

Locality gap is used to describe the gap between costs of the solution found and the absolute optimum solution. In other words, suppose S is a solution found by a local search algorithm L and O is the optimum solution. We say S has a locality gap of α if $\text{cost}(S) \geq \alpha\text{cost}(O)$.

1.3 Previous Work

The study of Budgeted Red-Blue Median from the perspective of approximation algorithms was initiated by Hajiaghayi, Khandekar, and Kortsarz [17] in 2012, where they considered a naive local search algorithm that takes in a feasible solution and iteratively improves the solution by swapping one red and/or blue facility until no significant improvement can be made. They were able to prove this is a constant factor approximation for Budgeted Red-Blue Median but they did not specify the constant. After showing a constant approximation for Budgeted Red-Blue Median, they continues to show the local search technique can also be applied to Prize Collecting k-Median and obtained a $(3+\epsilon)$-approximation [17]. In Prize Collecting k-Median, we have the option to not connect a client by paying a penalty. It is worth noting that the local search analysis improves the previously known bound of 4 by Charikar et al for the prize collecting version [9].

As mentioned before, the most special case of our problem is the well-known k-Median problem. Recall in k-Median, we are given n points in a metric space. We select at most k of these to be cluster centers and then assign each input point j to the selected center that is closest to it. If a point j is assigned to a center i, we incur a cost proportional to the distance between i and j. The goal is to select the k centers so as to minimize the total assignments cost. The first constant factor approximation for the k-Median problem was given by Charikar et al [8]. They used linear programming relaxation to round their solution and were able to obtain a $6\frac{2}{3}$ approximation.

Later this approximation guarantee was improved by Jain and Vazirani [19], they established a connection between k-Median and the Uncapacitated Facility Location Problem(UFL) such that k-Median is a Lagrangian relaxation of UFL. They were able to obtain a bi-point solution whose cost is at most 3 times the cost of an actual optimum solution using liner programming and rounding. A bi-point solution is a pair $\mathcal{F}_1, \mathcal{F}_2 \in \mathcal{F}$ such that $|\mathcal{F}_1| \leq k \leq |\mathcal{F}_2|$, along with reals $a, b \geq 0$ and $a + b = 1$ such that $a|\mathcal{F}_1| + b|\mathcal{F}_2| = k$. They showed a way to round such bi-point solution into an integral solution losing another factor of 2 so that in the end they get a 6 approximation [19]. Later, Jain, Mahdian and Saberi (JMS) improved the approximation ratio of finding the bi-point solution to 2 and thus resulting a 4 approximation for k-Median [18].

Following that, Arya et al [2] further improved the approximation ratio of k-Median to $(3 + \epsilon)$ using a local search based technique. To achieve this, given an instance of k-Median $\mathcal{I}$ and a set of facilities $\mathcal{F}$, let $S \in \mathcal{F}$ denote any solution output by the local search algorithm and let $O \in \mathcal{F}$ denote an optimum solution to $\mathcal{I}$. Let $i \in S, i^* \in O$, note that $(S - i + i^*)$ is a feasible solution because it opens the same amount of facilities as S. Examining the cost change in solution $(S - i + i^*)$ for all possible (i, i^*) pair, they were able to show the cost of S is at most 5 times the cost of O. By permitting up to a constant number p of facilities to be considered at the same time, they were able to reduce the bound to $3 + \frac{2}{p}$.

Later, Gupta et al [16] further simplifies the analysis of Arya et al [2] by using an averaging argument to avoid some complex anaylisis in the proofs of Arya et al. We use a randomized approach in our work that is fairly similar to and inspired by the averaging argument.

Very recently, Li and Svensson made a breakthrough in rounding the bi-point solution [22]. They first round a bi-point solution found by [18] into an integral solution with $k + O(1)$ open facilities while only losing a factor of $(\frac{1+\sqrt{3}}{2} + \epsilon)$ in the process. Such integral solutions can then be converted into an actual solution with only k open facilities. Combined with the factor 2 loss in finding the bi-points solution, they obtain a $(1+\sqrt{3}+\epsilon)$-approximation. Followed from there, Byrka et al [5] further improved rounding of a bi-point solution with a factor loss of $1.3371+\epsilon$ which is better than the previous $(\frac{1+\sqrt{(3)}}{2}+\epsilon) \sim 1.366+\epsilon$. Thus obtaining a $2.675+\epsilon$ approximation for k-MEDIAN.

It is also known that it is hard to approximate k-MEDIAN within a factor of $(1+\frac{2}{e^{\frac{1}{c}}})$ for any constant $0<c<1$ unless $P=NP$ [18].This is done through constructing a reduction from SET COVER. In SET COVER, we are given a set of items X and a set of sets S and the goal is to pick minimum number of sets whose union includes all the items.

Another stream of related problem is the MATROID MEDIAN PROBLEM. The MATROID MEDIAN PROBLEM is first introduced in [20] by Krishnaswamy et al and is a generalization of BUDGETED RED-BLUE MEDIAN. In the MATROID MEDIAN PROBLEM, we are given a set of facilities $\mathcal{F}$ and clients $\mathcal{C}$ with the constraint that the opened facilities forms an independent set from a matroid.

More generally, if we extend BUDGETED RED-BLUE MEDIAN to having t colors of facilities, this can be described as a MATROID MEDIAN PROBLEM instance with partition matroids with t parts which is known as the CONTENT DISTRIBUTION NETWORKS PROBLEM ([4], [17]). Krishnaswamy et al [20] obtained 16-approximation for MATROID MEDIAN and $O(1)$ approximation for the prize-collection variant of the problem.

A special case of MATROID MEDIAN is DATA PLACEMENT. Instead of facilities having different types, the clients now have different types of demand and each facility can provides certain types of good. The goal is to minimize the total cost of connecting clients to facilities that satisfy the demand constraints. This problem was studied by many, and a 10-approximation was given by Baev et al [3] using a linear program.

Later, Swamy improved the approximation ratio to 8 for MATROID MEDIAN problem and 24 for the prize-collecting version [26]. Swamy introduced a simpler way to round a natural linear program relaxation and completely avoided the two phase rounding procedure in the previous work.

Local search is one of the most powerful tools we have to solving various optimization problems. An early demonstration of local search technique in operation research dates back as far as 1963 where Kuehn et al [21] introduced first application of local search technique as a heuristics for solving UFL. In practice, local search is a powerful tool to obtain a feasible solution and is fast for many difficult problems such as Steiner Problems in graphs, Knapsack Problem, etc ([27], [2]). However due its context reliant nature, it is often challenging to analyze such algorithm.

One of the most well-known applications for local search is in the family of facility location problems. As mentioned before, local search achieved a $(3+\epsilon)$ approximation for k-Median by permitting simultaneously opening and closing a constant number of facilities and iteratively improving the solution cost. In other variations, Mahdian and Pál [23], and Svitkina and Tardos [25] consider settings where the opening cost of a facility is a function of the set of clients served by it. In [23], this cost is a nondecreasing function of the number of clients, and in [25] this cost arises from a certain tree defined on the client set. Devanur et al [11] and [16] consider k-Facility Location, which is similar to k-Median except that facilities also have opening costs. Gørtz and Nagarajan [15] considered a problem that they call the k-Median Forest, which generalizes k-Median, and obtained a $(3+\epsilon)$-approximation algorithm. Friggstad et al [1] considered Mobile facility location problem where instead of opening cost for facilities, we move the opening facilities with a movement cost. They obtained a $(3+\epsilon)$-approximation.

Very recently, local search is discovered to yield a *PTAS* for k-Means and k-Median in Euclidean space ([10], [12]). This is discovered independently by Cohen-Addad et al [10] and Friggstad et al [12] at the same time. In [12], they identified that local search yields a $(1+\epsilon)$approximation for k-Means in *doubling metric*. That is, the doubling dimension of a metric is defined to be the smallest d such that any ball of radius $2r$ can be covered with 2^d balls of radius r. If d is regarded to be a constant, the metric is then called a doubling metric. For example: constant-dimensional $\mathbb{R}^d$ under normal Euclidean distances or under manhattan distances is a doubling metric. Cohen and Addad also identifies that local search works also in "minor-closed" metrics, such as the shortest path metric of a planar graph [10].

Another well-known application for local search is approximating maximum independent set. Chan and Har-Peled presented an approximation for finding maximum independent set of pseudo disks [6]. A collection of simple (i.e. no holes) geometric objects are called pseudodisks if the boundaries of any two objects intersect in at most two points [6]. The algorithm produces a $PTAS$ using local search for the unweighted version of the problem and a constant factor approximation for the weighted version.

Mustafa and Ray [24] obtained a $(1 + \epsilon)$approximation for Geometric Hitting Set problem using local search. Note that this is discovered independent of Chan and Har-Peled. Later this is extended to Geometric Coverage Problem and a $PTAS$ is obtained by Chaplick et al [7]. Terrain Guarding is also a problem where local search technique is successfully applied and the first $PTAS$ is obtained by Gibson et al [14].

1.4 Thesis Outline

In Chapter 2 we take a closer look at a very tightly related problem k-Median. We give a brief discussion of the $(3 + \epsilon)$-approximation using the local search technique by Gupta et al [16] with a slight modification in the analysis to prepare for our Budgeted Red-Blue Median analysis. We will introduce the natural local search algorithm for k-Median. However the algorithm itself does not guarantee termination in polynomial number of iterations. We will present the standard workaround for this issue. We will also include some of the standard techniques and ideas typically used in our work.

We begin chapter 3 by giving an overview of the problem and preliminaries and notations. We then proceed to demonstrate how some of the techniques can be applied by analyzing a simple case of Budgeted Red-Blue Median. And then we dive into the full multi-swap analysis for the general case of Budgeted Red-Blue Median which is the main contribution of this work.

Next in Chapter 4, we examine the other end of the spectrum, showing the lower bounds for local search algorithm on BUDGETED RED-BLUE MEDIAN. Our approach also works for a more general setting where we allow a fixed number of colors for facilities instead of just red and blue. We show that for a fixed number t colors, the natural local search algorithm has a lower bound of $2t + 1$.

Finally, Chapter 5 summarizes the work of this thesis and gives our concluding remarks on the problems we explored. Open problems are presented which can be used to direct future work in this area. All new results appearing in this thesis were developed during the author's graduate studies. The results of Chapter 3 and Chapter 4 appeared in [13].

Chapter 2

Local Search Analysis for k-Median Problem

One problem that is closely related to Budgeted Red-Blue Median is k-Median. Aryan et al [2] proved that a multiple-swap local search algorithm is a $(3+\epsilon)$-approximation. Gupta and Tanwongsan [16] simplified this analysis. We present a modification of Gupta and Tanwongsan's analysis using the probabilistic method. This provides a gentle introduction to the multiple-swap analysis for Budgeted Red-Blue Median in Chapter 3.

2.1 Problem Definition and Preliminaries

Recall in k-Median, we are given a set of facilities $\mathcal{F}$ and a set of clients $\mathcal{C}$. The facilities and clients are scattered in a metric space with distances $d(u, v)$ between each $u, v \in \mathcal{F} \cup \mathcal{C}$. We are given a bound k and the goal is to open k facilities so that the sum of distances of every client to its nearest open facility is minimized. Here we describe the natural local search algorithm for k-Median, this was introduced by Arya et al:

Algorithm 1 The p-Swap local search algorithm for k-Median

```
Let $S$ be an arbitrary feasible solution.
while there is some feasible solution $S'$ with $|S - S'| \leq p$
        and cost($S'$) $<$ cost($S$) do
    $S \leftarrow S'$
end while
return $S$
```

A feasible solution is a set of facilities $S' \subseteq \mathcal{F}$ with $|S'| = k$. Since there is no reason to open less than k facilities, we can assume any feasible solution opens exactly k facilities. Algorithm 1 describes a procedure that iteratively improves a given feasible solution using a simple rule until no improvements can be found. The goal is to argue that any solution S produced by Algorithm 1 has a cost that is bounded by the cost of an optimum solution by some constant factor. In Chapter 3 we will introduce a similar local search algorithm for Budgeted Red-Blue Median. For convenience, from now on we will call any solution output by the local search algorithm the "locally optimum solution" and denote it by S; we will call the optimum solution to the problem the global optimum solution and denote it by O.

It is not clear that Algorithm 1 terminates in polynomial number of iteration. The standard workaround is trading accuracy for running time. With a slight modification, one can show that the number of iterations of the modified algorithm is strictly polynomial in the input size and only losing a $(1+\epsilon)$ multiplicative factor in the approximation guarantee. We will discuss the details of such technique at the end of this chapter.

Now we present the work of Gupta et al [16] with our modification and proving:

Theorem 2.1.1. *For any local optimum solution S,*

$$\text{cost}(S) \leq \left(3 + \frac{2}{p}\right) \text{cost}(O)$$

where O is a global optimum solution.

Let S be a local optimum solution and O be a global optimum solution, we can assume O and S are disjoint due to a simple duplicate argument, details of such argument can be found in [2]. Define $\phi : O \to S$ that maps each facility $i^* \in O$ to its closest facility $i \in S$. Naturally $\phi^{-1}(i)$ denotes $\{i^* \in O : \phi(i^*) = i\}$ and we let $\deg(i) = |\phi^{-1}(i)|$. We also introduce notation to describe various costs. We let o_j denote the nearest facility in O and s_j denote the nearest facility in S for client j. We also let $c_j = d(j, s_j), c_j^* = d(j, o_j)$.

Note that $\text{cost}(S) = \sum_{j \in C} c_j$, and $\text{cost}(O) = \sum_{j \in C} c_j^*$. Furthermore, for $i \in S$ we let $N(i)$ denote $\{j \in \mathcal{C} : s_j = i\}$, and for $i^* \in O$, let $N^*(i^*)$ denote $\{j \in \mathcal{C} : o_j = i^*\}$. The goal is to show that:

$$\text{cost}(S) \leq \left(3 + \frac{2}{p}\right) \text{cost}(O).$$

We now describe a way to partition both S and O. We will use this partition to describe the test swaps later.

Algorithm 2 Test swaps construction

```
$l = 0$
**while** there exists $r \in S$ such that $\deg(r) > 0$ **do**
    (1)$S_l = \{r\} \cup \{$ any $\deg(r) - 1$ facilities of S with degree 0$\}$
    (2)$O_l = \phi^{-1}(r)$
    (3)$S = S - S_l, O = O - O_l, l = l + 1$
**end while**
**return** $\{S_l\}_{l=1}^L, \{O_l\}_{l=1}^L$
```

Algorithm 2 terminates with two sequences $\{S_l\}_{l=1}^L$ and $\{O_l\}_{l=1}^L$. For any $1 \leq l \leq L$ we have $|S_l| = |O_l|$. We need to argue that Algorithm 2 terminates as expected and produces the desired sequences.The key idea is that there is always enough facilities with degree 0 in S to execute step (1) in the while loop and this can be shown by a simple counting argument. The following is also helpful.

Lemma 2.1.2. *If $s_j \in S_l$ and $o_j \notin O_l$, then $\phi(o_j) \notin S_l$*

Proof. Let j be a client and $s_j \in S_l$ and $o_j \notin O_l$ for some index l. Suppose $\phi(o_j) \in S_l$, this implies $\deg(\phi(o_j)) > 0$. From the step (2) in Algorithm 2 it must be the case that $O_l = \phi^{-1}(\phi(o_j))$. This implies $o_j \in O_l$ which is a contradiction thus proves the lemma. $\square$

The following lemma is helpful for analyzing distance:

Lemma 2.1.3. *For any $j \in \mathcal{C}$, $d(j, \phi(o_j)) - c_j \leq 2c_j^*$.*

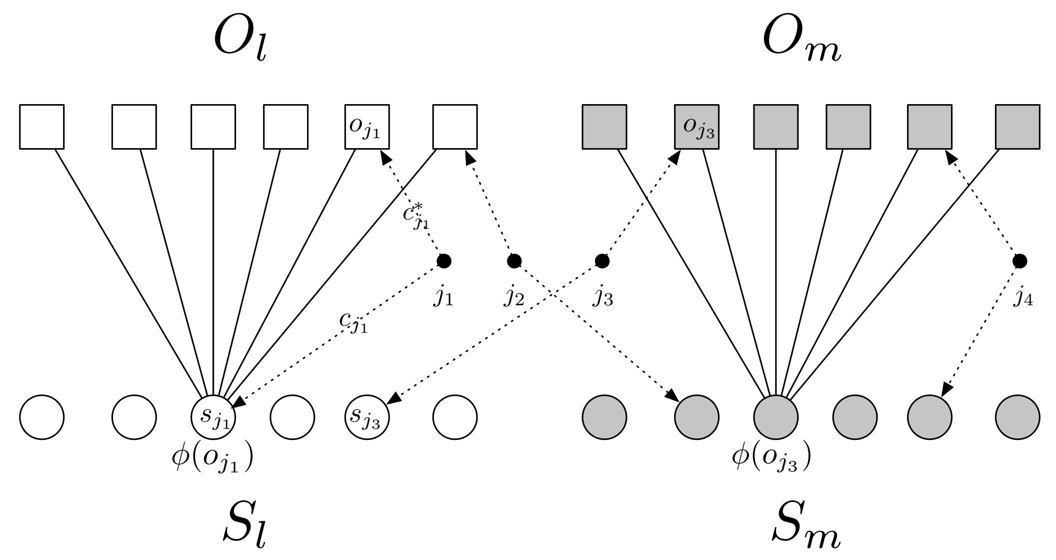

Figure 2.1: Example of a test swap. White square and circles represents a part S_l, O_l that is being swapped and the grey squares and circle represents another part O_m, S_m that is not being swapped. clients $j_1, j_2 \in N^*(O_l)$; client $j_3 \in N(S_l) \backslash N^*(O_l)$.

Proof. By the triangle inequality and the definition of ϕ, we have

$$d(j, \phi(o_j)) \leq d(j, o_j) + d(o_j, \phi(o_j)) \leq c^*_j + d(o_j, s_j) \leq 2c^*_j + c_j.$$

□

Now we describe a series of test swaps for each part $S_l \cup O_l$ of the partition:

Lemma 2.1.4. *For parts S_l, O_l with $|S_l| = |O_l| \leq p$:*

$$0 \leq \sum_{j \in N^*(O_l)} (c^*_j - c_j) + \sum_{j \in N(S_l)} 2c_j.$$

Proof. For any locally optimum solution S, if we swap at most p facilities with $\mathcal{F} \backslash S$, then the cost change is guaranteed to be non-negative. Because $|S_l| = |O_l| \leq p$, we get $0 \leq \text{cost}(S - S_l + O_l) - \text{cost}(S)$.

We describe a way to reassign clients to upper bound the cost change. The following client reassignment plan ensures every client is assigned to an open facility:

1. For all clients $j \in N^*(O_l)$, assign j to o_j.

2. For all clients $j \in N(S_l) \backslash N^*(O_l)$, assign j to $\phi(o_j)$.

3. The remaining clients assign to s_j.

This assignment is valid because we did not assign any clients to a closed facility due to Lemma 2.1.2. Take an example of Figure 2.1. O_l and S_l is being considered for a test swap, O_m, S_m is not. This implies after the test swap, white squares will be open, and the grey circle $\phi(o_{j_3})$ will also be open because S_m is not being swapped out. Clients j_1, j_2 belongs to the case 1; client j_3 belongs to case 2; and client j_4 belongs to case 3. Figure 2.1 gives an illustration of different types of clients whose cost contribution needs to be carefully considered before and after a test swap. We now give the cost change analysis for these clients.

It is obvious to see from 1, the cost change for clients $j \in N^*(O_l)$ is $d(j, o_j) - d(j, s_j) = c_j^* - c_j$. From 2, we get the cost change for clients $j \in N(S_l) \backslash N^*(O_l) = d(j, \phi(o_j)) - d(j, s_j)$. Apply Lemma 2.1.3 and we get $d(j, \phi(o_j)) - d(j, s_j) \leq 2c_j^*$. From 3, the cost change for the rest of the clients is 0. Because $\sum_{j \in N(S_l) \backslash N^*(O_l)} 2c_j \leq \sum_{j \in N(S_l)} 2c_j$, and we described a valid reassignment which may not be the best reassignment but it is sufficient for proving the lemma. □

The following proof is where we differ from the Gupta and Tanwongsan analysis slightly. They perform many more swaps and average their resulting inequalities. We perform fewer swaps, but they are randomly chosen. Essentially we achieve the same inequality but we chose the probabilistic method in our Budgeted Red-Blue Median analysis.

Lemma 2.1.5. *For parts $|S_l| = |O_l| = t > p$:*

$$0 \leq \sum_{j \in N^*(O_l)} (c_j^* - c_j) + \left(1 + \frac{1}{p}\right) \sum_{j \in N(S_l)} 2c_j^*.$$

Proof. This proof generally follows the proof of Lemma 2.1.4, however the size of the partition we consider is $k > p$. So if we were to swap out S_l and swap O_l entirely, we are not guaranteed the overall cost change is non-negative. Here we construct a different set of test swaps for this partition.

Let $r \in S_l$ be the facility with $\deg(r) > 0$. Let $\hat{S}_l := S_l - r$. Since there is exactly one facility $r \in S_l$ with $\deg(r) > 0$, we are guaranteed all $i \in \hat{S}_l$ have $\deg(i) = 0$. Now consider the following test swaps: For each facility $i^* \in O_l$, pick a random facility $i \in \hat{S}_l$ and swap the pair (i, i^*). Note that we swap only one pair of facilities at a time therefore the cost does not decrease.

For each test swap (i, i^*), consider the following client reassignment plan:

1. For all clients $j \in N^*(i^*)$, move j to i^*

2. For all clients $j \in N(i) \backslash N^*(i^*)$, move j to $\phi(o_j)$

3. The remaining clients j assign to s_j.

All clients $j \in N^*(i^*)$ have their cost change exactly $c_j^* - c_j$. From (2), because $\deg(i) = 0$ for any $i \in \hat{S}_l$, we are guaranteed $\phi(o_j) \neq i$ therefore $\phi(o_j)$ must be open. Now the cost change for these clients are again: $d(j, \phi(o_j)) - d(j, s_j) \leq 2c_j^*$ by Lemma 2.1.3. Lastly from (3), the cost change for the rest of the clients is 0.

Now we average these bounds over the random choices for the swaps. Consider the randomly chosen $i \in \hat{S}_l$ for each test swap, the probability that a specific i is chosen is $\frac{1}{|\hat{S}_l|} = \frac{1}{t-1}$. So averaging over all possible test swaps for a fixed i^* and a randomly chosen i we get the expected cost change for each test swap is:

$$0 \leq \sum_{j \in N^*(i^*)} (c_j^* - c_j) + \frac{1}{t-1} \sum_{j \in N(\hat{S}_l) \backslash N^*(O_l)} 2c_j.$$

Note that $\sum_{j \in N(\hat{S}_l) \backslash N^*(O_l)} 2c_j \leq \sum_{j \in N(S_l)} 2c_j$ so we take it as an upper bound, summing over all test swaps for all $i^* \in O_l$, we get:

$$0 \leq \sum_{j \in N^*(O_l)} (c_j^* - c_j) + \frac{t}{t-1} \sum_{j \in N(S_l)} 2c_j.$$

Finally, note $\frac{t}{t-1} \leq 1 + \frac{1}{p}$ because $t \geq p + 1$. Lemma 2.1.5 follows.

□

We conclude by combining Lemma 2.1.3 and 2.1.5, summing the inequalities over all parts $S_l \cup O_l$ and recalling that the parts $\{S_l \cup O_l\}$ form a partition of $S \cup O$, we see:

$$0 \leq \sum_{j \in \mathcal{C}} (c_j^* - c_j) + (1 + \frac{1}{p}) \sum_{j \in \mathcal{C}} 2c_j^*$$

This proves Theorem 2.1.1.

Polynomial-time Adaptation of Algorithm 1

As mentioned before, it is not clear Algorithm 1 runs in polynomial time. Here we show a standard technique adaptation of Algorithm 1 to guarantee the following:

Theorem 2.1.6. *Algorithm 3 has an execution time that is polynomial in the inputs size, and produces a* $\left(\frac{3+\frac{2}{p}}{1-\epsilon}\right)$*-approximation to the k-Median problem for some constant* ϵ*.*

Algorithm 3 A polynomial time local search algorithm for k-Median

```
Let $S$ be arbitrary feasible solution.
$\epsilon \leftarrow$ constant
while there is some feasible solution $S'$ with $|S - S'| \leq p$
        and cost($S'$) $< (1 - \frac{\epsilon}{k})$cost($S$) do
    $S \leftarrow S'$
end while
return $S$
```

Proof. We first show that Algorithm 3 runs in polynomial time.

At each iteration of the step 2 of Algorithm 3 there will be at most polynomial number of feasible solution S' candidate to check before executing step 2. This is because we consider up to p number of facilities to close and open each iteration so the number of candidate solution S' is upper bounded by $\binom{n}{p}^2 \leq n^{2p}$ for a constant p where $|\mathcal{F}| = n$. In addition, evaluating cost(S') is polynomial in $|\mathcal{C}|$ and upper bounded by $p|\mathcal{C}|$ since the cost of a solution is only dependant on the distance of clients to their closest open facility. Therefore finding a feasible candidate S' for each iteration only takes polynomial time of the input size.

On the other hand, note that each iteration we guarantee to improve the solution by at least $(\frac{\epsilon}{k})\text{cost}(S)$. If we let S_0 denotes the initial input feasible solution and let S^* denotes the solution with minimum non-zero cost then the number of iterations is at most $\log\left(\text{cost}(S_0)/\text{cost}(S^*)\right)/\log\left(\frac{1}{1-\frac{\epsilon}{k}}\right)+1$. Note that S^* is either the global OPT or the global OPT has a 0 cost. Now $\text{cost}(S_0) \leq l|\mathcal{C}|$ where l is the longest distance between a client and a facility and $\text{cost}(S^*)$ is at least the shortest non-zero distance between a client and a facility. So the number of bits to represent $\log\left(\left(l|\mathcal{C}|\right)/\text{cost}\left(S^*\right)\right)$ is polynomially bounded by the input size. Thus the number of iterations is also polynomial in input. Therefore Algorithm 3 terminates in polynomial time.

Let O be the global optimum solution and S' be the output of Algorithm 3. Now recall the analysis from the previous section, following Lemma 2.1.3 and Lemma 2.1.5, change the lower bound on the cost change for each inequality from 0 to $-\frac{\epsilon}{k}\text{cost}(S')$. The rest of the analysis follows for Algorithm 3. We sum the cost change over all parts again, we get:

$$-\sum_{l=1}^{L}\frac{\epsilon}{k}\sum_{j\in\mathcal{C}}c_j \leq \sum_{j\in\mathcal{C}}(c_j^*-c_j)+(1+\frac{1}{p})\sum_{j\in\mathcal{C}}2c_j^*.$$

Clearly $L \leq k$, we have:

$$(1-\epsilon)\sum_{j\in C}c_j \leq (3+\frac{2}{p})\sum_{j\in C_j^*}$$

which proves Theorem 2.1.6.

$\square$

Chapter 3

The Analysis of Budgeted Red-Blue Median

3.1 Overview and the Challenges

We begin this chapter by recalling the precise definition of our problem. In Budgeted Red-Blue Median, we are given a set of clients $\mathcal{C}$ and two sets of facilities, namely red facilities $\mathcal{R}$ and blue facilities $\mathcal{B}$. These locations are within a metric space with metric distance $d(i, j) \geq 0$ for any two $i, j \in C \cup \mathcal{R} \cup \mathcal{B}$. Additionally we are given two budgets k_r, k_b for the two types of facilities. The goal is to select a subset of facilities $R \subseteq \mathcal{R}, B \subseteq \mathcal{B}$ with $|R| = k_r, |B| = k_b$ while minimizing

$$\mathrm{cost}(R \cup B) = \sum_{j \in C} \min_{i \in R \cup B} d(i, j).$$

A feasible solution is a subset of facilities $R \subseteq \mathcal{R}, B \subseteq \mathcal{B}$ with $|R| = k_r, |B| = k_b$. A natural local search algorithm starts with an arbitrary feasible solution and iteratively improves the solution by changing the selection of facilities up to a constant number of facilities until no such improvement is possible.

Algorithm 4 is a generalization of local search algorithm for k-Median from Chapter 2. As with the k-Median local search algorithm in Chapter 2, it is not necessarily true this algorithm runs in polynomial time. The standard workaround will be presented later. The main idea is to only make the improvement in each iteration when the improvement is significant enough. This

Algorithm 4 The p-Swap Heuristic for Budgeted Red Blue Median

```
Let $(R, B)$ be arbitrary feasible solution and **constant** $p$.
**while** there is some feasible solution $(R', B')$ with $|R - R'| \leq p$
        and $|B - B'| \leq p$ and $\text{cost}(R' \cup B') < \text{cost}(R \cup B)$ **do**
    $(R, B) \leftarrow (R', B')$
**end while**
**return** $(R, B)$
```

simple detail is shown at the end of the Chapter. We show that Algorithm 4 achieves the following:

Theorem 3.1.1. *A locally optimum solution $S = R \cup B$ returned by Algorithm 4 has* $\text{cost}(S) \leq \left(5 + O(\frac{1}{\sqrt{p}})\right) \text{cost}(O)$ *where O is a global optimum solution.*

3.1.1 Overview of our technique

Before we actually prove Theorem 3.1.1, we briefly recall the techniques from Chapter 2 and discuss how they will be adapted in the coming proof.

We first emphasize the separation of theoretical analysis and the actual local search algorithm. The algorithm itself simply tries small local changes to try and find improved solutions. The analysis itself involves more intricate steps, namely analyzing test swaps, but the reader should remember this is only to generate the required inequalities to prove Theorem 3.1.1; these steps are not part of the actual algorithm.

Recall in the proof of Theorem 2.1.1 we used test swaps to generate some inequalities that are used in the final bound. Intuitively, a test swap attempts to make a change to a locally optimum solution. As long as the number of facilities being swapped is bounded by the size of the swaps of the algorithm, we know the cost of the resulting solution is no cheaper than the local optimum solution cost. An explicit upper bound was made on the cost change by describing how each client could be reassigned.

The test swaps in the proof of Theorem 2.1.1 were generated by focussing on blocks. That is, the analysis first partitioned the facilities in the local and global optimum solutions into blocks. In turn, the analysis focussed on each block and generated a collection of test swaps for the block. Combining the

the corresponding inequalities over all swaps generated for all blocks revealed enough information to bound the cost of the local optimum cost by a constant factor of the global optimum cost.

The proof of Theorem 3.1.1 follows the same approach, but each step is more involved. Not only do we require each block to be balanced between the local and global optimum, each colour must also be balanced and some additional properties listed later are required to complete our analysis. So our block partitioning scheme is more delicate.

The test swaps are motivated the same way as in Theorem 2.1.1; we want to ensure each facility in the global optimum is opened once to ensure we get "negative dependence" on the local optimum cost for each client. This requires a careful selection of which local optimum facilities to close in the swaps, and the chief difficulty is in reassigning the clients served by these facilities in a relatively cheap manner. To complicate things further, we are required to open some global optimum facilities in more than one test swap to compensate for some positive dependence on local optimum service cost for some clients.

Ultimately, we still generate a single inequality for each block that bounds the local optimum service cost for some clients against a constant factor times the global optimum service cost for some clients. Adding these inequalities for all blocks will complete the proof Theorem 3.1.1.

Our final bound of $5 + \epsilon$ is worse than the guarantee given in Theorem 2.1.1. In Chapter 4, we show this is unavoidable. Namely, that we cannot guarantee better than a 5-approximation for any constant-size neighbourhood to swap.

3.1.2 Preliminaries

We define some notation here to help us illustrate our techniques. Let $S = R \cup B$ with $R \in \mathcal{R}$ and $B \in \mathcal{B}$ be a locally optimum solution obtained by Algorithm 4. We also fix a global optimum solution $O = R^* \cup B^*$ where $R^* \subseteq \mathcal{R}$ and $B^* \subseteq \mathcal{B}$. Intuitively, we want to somehow assess the quality of S. The way we achieve this is by constructing test swaps between S and O to establish some relations between $\text{cost}(S)$ and $\text{cost}(O)$, much like the analysis of Algorithm 1 from Chapter 2. We can assume $O \cap S = \emptyset$, otherwise we can just duplicate a facility at the same location and add one copy to each of the O and S. Such change ensures S still maintains a locally optimum solution. This can be easily verified.

For any client $j \in C$ we let $s_j \in S$ be the closest facility to j in the local optimum and $o_j \in O$ be the closest facility to j in the global optimum. For clarity, we let $c_j = d(j, s_j)$ and $c_j^* = d(j, o_j)$ denote the cost contribution respectively. Thus $\text{cost}(S) = \sum_{j \in C} c_j$ and $\text{cost}(O) = \sum_{j \in O} c_j^*$. For convenience, we let $N(i) = \{j \in C : s_j = i\}$ for any $i \in S$ and $N^*(i^*) = \{j \in C : o_j = i^*\}$ for any $i^* \in O$.

Let $\phi : O \to S$ map each facility in O to its nearest facility in S, breaking ties arbitrarily. For $i \in S$, let $\deg(i) = |\phi^{-1}(i)|$. If $\deg(i) \neq 0$, let $\text{cent}(i)$ be the facility in $\phi^{-1}(i)$ that is closest to i, again breaking ties arbitrarily.

We borrow a definition from [17]:

Definition 3.1.2 (very good, good, bad facility)**.** *A facility $i \in S$ is very good if $\deg(i) = 0$, good if no $i^* \in \phi^{-1}(i)$ has the same colour as i, and bad otherwise.*

This definition will help us to partition the set of facilities. Recall the partitioning procedure introduced in Chapter 2, our work requires partitioning the set of facilities into blocks with slight variation. The major differences here are that we are additionally required to balance the two colours in each block and the blocks are also allowed to have more than one facility i with $\deg(i) > 0$. We give the following lemma to describe blocks:

Lemma 3.1.3. *We can partition $S \cup O$ into* blocks *T satisfying the following properties.*

- $|T \cap R| = |T \cap R^*|$ *and* $|T \cap B| = |T \cap B^*|$.
- *For every $i \in S \cap T$, we also have $\phi^{-1}(i) \subseteq T$. For every $i^* \in O \cap T$, we have $\phi(i^*) \in T$.*

- *There is some facility* $\hat{i} \in T \cap S$ *with* $\deg(\hat{i}) > 0$ *designated as the* leader *that has the following properties. Every other* $i \in T \cap S - \{\hat{i}\}$ *is either good or very good and all good* $i \in T \cap S - \{\hat{i}\}$ *have the same colour.*

The blocks are helpful structures we use in our analysis. Intuitively, each block is constructed around each individual bad facility so that each block contains exactly one bad facility until there is no more bad facilities left, then blocks are formed with the leftover facilities. Figure 3.2 in Section 3.4 shows an example of a block. The details of generating blocks are discussed in Section 3.3.

As before, we often consider operations that add or remove a single item from a set. To keep the notation cleaner, we let $S + i$ and $S - i$ refer to $S \cup \{i\}$ and $S - \{i\}$, respectively, for sets S and items i. We also say *swap in* a facility to mean opening a facility and *swap out* a facility as closing a facility.

In the next section, we give a detailed analysis of a simple example of Budgeted Red-Blue Median. In Section 3.3 we give details about block generations. Then finally, we dive into the full multi-swap analysis of the Budgeted Red-Blue Median in the last section.

3.2 A brief warm up

Hajiaghayi et al [17] gives an introduction of the first constant approximation for Budgeted Red-Blue Median. Though they did not specify the constant. We note that Hajiaghayi et al [17] essentially analyzed Algorithm 4 with the input parameter $p = 1$. In the discussion of lower bounds for Algorithm 4 in Chapter 4 we show that Algorithm 4 with parameter $p = 1$ has a lower bound of 7 in that there are solutions that would not be improved by the local search step with $p = 1$ yet are as bad as $7 - \epsilon$ times the optimum for arbitrarily small ϵ.

Recall that $\text{cost}(S) = \sum_{j \in C} c_j$ and $\text{cost}(O) = \sum_{j \in C} c^*_j$. The goal is to show $\sum_{j \in C} c_j \leq \alpha \sum_{j \in C} c^*_j$ for some constant α so then we have an α-approximation for Budgeted Red-Blue Median. We will now illustrate some challenges we face when migrating the standard analysis of local search on k-Median onto Budgeted Red-Blue Median using an simple instance of the Budgeted Red-Blue Median.

In Figure 3.1 a simple instance of Budgeted Red-Blue Median is given. In this instance we consider a local search algorithm with single swap heuristics, that is we are only permitted to construct test swaps involving at most one pair of blue and/or red facilities. We set the size of the instance to be $|S| = |O| = h + 1$ with $|B| = |B^*| = 1$ and $|R| = |R^*| = h$ for a large constant h. Since we only have 1 blue facility $i^* \in O$, then whenever a test swap swaps in i^* we are forced to swap out $i \in B$. However since $\deg(i)$ is large and each test swap only permits swapping one pair of blue and/or red facilities, then all clients $j \in N^*(\phi^{-1}(i))$ need to be reassigned to some open facility. We cannot assign these clients to i^* because $d(i, i^*)$ is unbounded. To work around this issue we introduce the following procedure, note that this is only one way of constructing test swaps that suited particularly for this simple setting, we do not guarantee these test swaps generalize in the general setting:

Step 1: **Swap in:** i^* , cent(i) **Swap out:** i , i_a for a random $a \in \{1, 2, 3, ..., h\}$

Step 2: For each $k \in \{1, 2, 3, ..., h\} - \text{cent}(i)$: **Swap in:** ${i_k}^*$ **Swap out:** i_a for a random $a \in \{1, 2, 3, ..., h\}$

Step 3: Repeat Step 2 once.

Lemma 3.2.1. *For any $j \in C$, $d(j, \phi(o_j)) - c_j \leq 2c_j^*$.*

The proof for Lemma 2.1.4 works here as well therefore we omit the proof here. Another useful lemma for analyzing the cost is following:

Lemma 3.2.2. *For any $j \in C$, $d(j, \text{cent}(\phi(o_j))) - c_j \leq 3c_j^* + c_j$.*

Proof. By the triangle inequality and Lemma 3.2.1, it suffices to prove

$$d(\phi(o_j), \text{cent}(\phi(o_j))) \leq c_j^* + c_j.$$

By definition of cent() and ϕ,

$$d(\phi(o_j), \text{cent}(\phi(o_j))) \leq d(\phi(o_j), o_j) \leq d(s_j, o_j) \leq c_j^* + c_j.$$

□

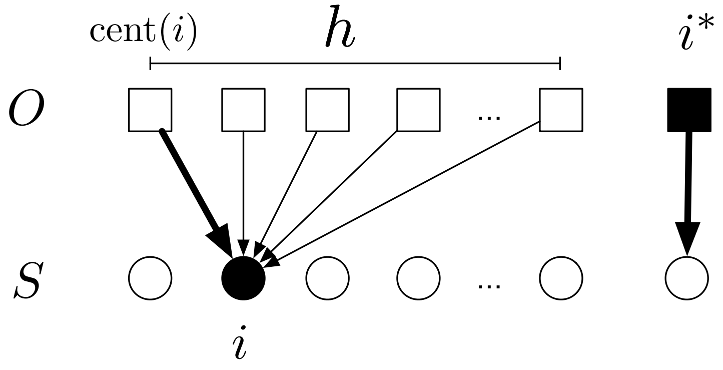

Figure 3.1: Illustration of a simple instance of the problem. In this instance, we let squares denotes O and circles denotes S. We also have black represents $\mathcal{B}$ and white represents $\mathcal{R}$. The solid lines depicts ϕ and the thick line connects cent(i) to i.

Now we give the details on analyzing a simple Red Blue Median instance. This provides a gentle introduction to our analysis for the general setting of Budgeted Red-Blue Median in Section 3.4. To summerize:

Theorem 3.2.3. *Figure 3.1 illustrates* Budgeted Red-Blue Median *with a locally optimum solution S from single swap heuristic such that*

$$\text{cost}(S) \leq \left(5 + \frac{4}{h-1}\right) \text{cost}(O)$$

when O is a global optimum solution.

Proof.

Step 1:

Consider the Step 1 of the candidate test swaps. We swap one pair of red facilities $(\text{cent}(i), i_a)$ and one pair of blue facilities (i^*, i) at the same time. We note that the cost change is non-negative because S is a local optimum solution and the test swap only consider up to one pairs of facilities. For clients $j \in N^*(i^*) \cup N^*(\text{cent}(i))$, we just send them to o_j and bound their cost change by $c^*_j - c_j$. Next note that only i^* and cent(i) are open after the swaps in step 1, so for all clients j either $o_j = i^*$ or $\phi(o_j)$ is closed. Now consider clients $j \in N(i) \backslash (N^*(i^*) \cup N^*(\text{cent}(i)))$, we assign these clients to cent(i). The cost of reassigning these clients are $d(\text{cent}(i), j) - d(j, i) \leq d(\text{cent}(i), i)$ by the triangle inequality. Because $d(\text{cent}(i), i) \leq d(o_j, i)$ by definition of cent(), we get the cost change is $d(o_j, i) \leq d(o_j, j) + d(j, s_j) = c_j^* + c_j$.

Finally, for clients $j \in N(i_a) \backslash (N^*(i^*) \cup N^*(\text{cent}(i)))$, because both o_j and $\phi(o_j)$ are closed, we have to assign these clients to cent(i). Applying Lemma 3.2.2 we get the cost change is $3c_j^* + c_j$. Now i_a is randomly chosen, so the probability of a client belong to $N(i_a)$ is $\frac{1}{h}$. Therefore the expected cost change of a single swap here is $\frac{1}{h}\sum_{j \in N(R)}(3c_j^* + c_j)$. Summing over all clients we get:

$$0 \leq \sum_{j \in N^*(i^*) \cup N^*(\text{cent}(i))} (c_j^* - c_j) + \sum_{j \in N(i) \backslash (N^*(i^*) \cup N^*(\text{cent}(i)))} (c_j^* + c_j) + \frac{1}{h} \sum_{j \in N(R)} (3c_j^* + c_j).$$

Note that we generated some positive c_j terms for some clients. The Step 3 is to generate some extra negative c_j terms to cancel out the positive c_j terms so that we have one copy of $-c_j$ terms of each client in the end.

Step 2:

Now we consider Step 2. For each of the $({i_k}^*, i_a)$ swap, we assign j to o_j and get $c_j^* - c_j$ for every clients $j \in N^*({i_k}^*)$. For clients $j \in N(i_a) \backslash N^*({i_k}^*)$, since $\deg(i_a) = 0$ we can assign j to $\phi(o_j)$ and get the cost change of $2c_j^*$ using Lemma 3.2.1. Then the expected cost change for each individual swap is $\frac{1}{h}\sum_{j \in C} 2c_j^*$. Finally combining all $h - 1$ inequalities we get:

$$0 \leq \sum_{j \in N^*(R^*) \backslash N^*(\text{cent}(i))} (c_j^* - c_j) + \frac{h-1}{h} \sum_{j \in N(R)} 2c_j^*.$$

Step 3:

Step 3 is similar to step 2, but we only reroute a client j to o_j if $j \in N(i) \backslash (N^*(i^*) \cup N^*(\text{cent}(i)))$. Thus we get another copy of $c_j^* - c_j$ for these clients. Also i is open so we assign every clients $j \in N(i_a)$ to i and thus causing a cost change of $2c^*_j$ for them. Again, with probability of $\frac{1}{h}$, j belongs to $N(i_a)$, we conclude the expected cost change is $\frac{1}{h}\sum_{j\in R} 2c^*_j$. Combining all cost change from the $h-1$ test swaps we have :

$$0 \leq \sum_{j\in N(i)\setminus(N^*(i^*)\cup N^*(\text{cent}(i)))} (c^*_j - c_j) + \frac{h-1}{h}\sum_{j\in N(R)} 2c^*_j.$$

Combining all three inequalities we get:

$$\frac{h-1}{h}\sum_{j\in C} c_j \leq \sum_{j\in N(i)\cap(N^*(i^*)\cup N^*(\text{cent}(i)))} c^*_j + \sum_{j\in N(i)\setminus(N^*(i^*)\cup N^*(\text{cent}(i)))} 3c^*_j + \frac{5h-1}{h}\sum_{j\in N(R)} c^*_j.$$

Now summarizing:

$$\sum_{j\in C} c_j \leq \left(5 + \frac{4}{h-1}\right)\sum_{j\in C} c^*_j$$

$\square$

From Theorem 3.2.3 we noted that for a large h, we get $(5+\epsilon)$approximation for this simple instance of BUDGETED RED-BLUE MEDIAN.

3.3 Block Generation

In this section we give details about our procedure for generating blocks.

Recall Lemma 3.1.3 for the description of a block T:

- $|T \cap R| = |T \cap R^*|$ and $|T \cap B| = |T \cap B^*|$.
- For every $i \in S \cap T$, we also have $\phi^{-1}(i) \subseteq T$. For every $i^* \in O \cap T$, we have $\phi(i^*) \in T$.
- There is some facility $\hat{i} \in T \cap S$ with $\deg(\hat{i}) > 0$ designated as the *leader* that has the following properties. Every other $i \in T \cap S - \{\hat{i}\}$ is either good or very good and all good $i \in T \cap S - \{\hat{i}\}$ have the same colour.

For convenience, we briefly recall the classification of facilities $i \in S$ from Definition 3.1.2.

- **Very Good**: $\phi^{-1}(i) = \emptyset$.
- **Good**: $\phi^{-1}(i) \neq \emptyset$ but no facility in $\phi^{-1}(i)$ has the same colour as i.
- **Bad**: i is neither very good nor good. Some facility in $\phi^{-1}(i)$ has the same colour as i.

Before creating the blocks, we first describe how to partition $S \cup O$ into groups. These groups will then be combined to form the final blocks. We say that a *group* is a subset G of $S \cup O$ where $|G \cap S| = |G \cap O|$, there is exactly one $\hat{i} \in G \cap S$ with $\deg(\hat{i}) > 0$, and $G \cap O = \phi^{-1}(\hat{i})$. Call this facility $\hat{i}$ the *representative* of the group.

We classify groups G in one of three ways.

- **Balanced**: $|G \cap R| = |G \cap R^*|$ and $|G \cap B| = |G \cap B^*|$.
- **Good**: $\hat{i}$ is a good facility and all other $i \in G \cap S - \hat{i}$ have a different colour than $\hat{i}$. Note this means $|G \cap R| = |G \cap R^*| \pm 1$.
- **Bad**: G is neither balanced nor good.

Algorithm 5 describes a procedure for forming groups in a particular way that will be helpful in creating the final blocks.

Algorithm 5 Procedure for partitioning into groups

```
$S' \leftarrow S, O' \leftarrow O$
**while** $\exists$ some facility $i$ in $S'$ with $\deg(i) > 0$ **do**
    $G \leftarrow \phi^{-1}(i) + i$
    **if** $G \cup X$ is a balanced group for some $X \subseteq S'$ **then**
        $G' \leftarrow G \cup X$
    **else if** $G \cup X$ is a good group for some $X \subseteq S'$ **then**
        $G' \leftarrow G \cup X$
    **else**
        Let $X \subseteq \{i' \in S' : \deg(i') = 0\}$ such that $G \cup X$ is a bad group and
        all facilities in $S' - X$ have the same colour.        ▷ c.f. Lemma 3.3.1
        $G' \leftarrow G \cup X$.
    **end if**
    Output group $G'$                                  ▷ $i$ is the representative of $G'$
    $S' \leftarrow S' - G', O' \leftarrow O' - G'$
**end while**
```

Lemma 3.3.1. *Algorithm 5 partitions $S \cup O$ into a set of groups.*

Proof. We first show that there are always exactly

$$|O'| - |\{i \in S' : \deg(i) \neq 0\}|$$

very good facilities in S'. During the execution of Algorithm 5, let set $\hat{S} = \{i \in S' : \deg(i) \neq 0\}$. The size of O' is $|O'| = \sum_{i \in \hat{S}} |\phi^{-1}(i)|$ because every facility $i^* \in O$ must belongs to $\phi^{-1}(i)$ for some $i \in \hat{S}$, and $\phi^{-1}(i') \cap \phi^{-1}(i'') = \emptyset$ for any $i', i'' \in \hat{S}, i' \neq i''$ by the definition of $\phi()$. Every other facility in $i \in S \backslash \hat{S}$ has $\deg(i) = 0$ so the number of facilities with degree 0 is exactly $|O'| - |\hat{S}| = |O'| - |\{i \in S' : \deg(i) \neq 0\}|$.

Thus we can always find a subset of very good facilities X such that $G \cup X$ is a group with $|O \cap \{G \cup X\}| = |S \cap \{G \cup X\}|$. We prove that if we can not find a set $G \cup X$ that is either a balanced group or good group, then we can find X to ensure $S' - X$ only contains facilities of one colour.

There are 2 cases. Suppose i is bad and, without loss of generality, that it is also red. Because we cannot extend G to be a balanced group, either there are less than $|\phi^{-1}(i) \cap \mathcal{B}|$ very good blue facilities in S' or less than $|\phi^{-1}(i) \cap \mathcal{R}| - 1$ very good red facilities in S'. In either case, first add all very good facilities from the "deficient" colour to X to use up that colour and then add enough very good facilities to X of the other colour to ensure $|X| = |\phi^{-1}(i)| - 1$.

In the other case when i is good, we again assume without loss of generality that it is red. Because we cannot form a good group, there are fewer than $|\phi^{-1}(i)| - 1$ very good blue facilities in S'. Use them up when forming X and then add enough very good red facilities so that $|X| = |\phi^{-1}(i)| - 1$. □

Let $\mathcal{G}$ be the collection of groups output by Algorithm 5. We now show how to piece these groups together to form blocks.

It is easy to verify that any "block" that is output by this algorithm indeed satisfies the properties listed in Lemma 3.1.3.

The following lemma explains why this procedure correctly executes and why all groups are used up. That is, it finishes the partitioning of $S \cup O$ into blocks. For a union of groups $G^* = G_1 \cup \ldots \cup G_k$, define the *blue deficiency* of G^* as $|G^* \cap B^*| - |G^* \cap B|$.

Algorithm 6 Procedure for generating blocks

```
$\mathcal{G}' \leftarrow \mathcal{G}$
**while** there is some balanced group $G$ in $\mathcal{G}'$ **do**
    Output $G$ as a block with its own representative being the leader.
    $\mathcal{G}' \leftarrow \mathcal{G}' - G$.
**end while**
**while** there are good groups $G, G' \in \mathcal{G}'$ with different coloured representatives **do**
    Output the block $G \cup G'$ and choose either representative as the leader.
    $\mathcal{G}' \leftarrow \mathcal{G}' - \{G, G'\}$
**end while**
**while** there is some bad group $G \in \mathcal{G}'$ **do**                ▷ c.f. Lemma 3.3.2.
    Let $\mathcal{G}_0 \subseteq \mathcal{G}' - G$ consist only of good groups such that $G + \mathcal{G}_0$ is a block.
    Output $G + \mathcal{G}_0$ with the representative of $G$ as the leader.
    $\mathcal{G}' \leftarrow \mathcal{G}' - (G + \mathcal{G}_0)$.
**end while**
```

Lemma 3.3.2. *If G is a bad group considered in some iteration of the last loop, we can find the corresponding $\mathcal{G}_0$ so that $G + \mathcal{G}_0$ is a block. Furthermore, after the last while loop terminates then $\mathcal{G}' = \emptyset$.*

Proof. Suppose, without loss of generality, that the blue very good facilities were used up the first time a bad group was formed in Algorithm 5. Thus, for every bad group $G' \in \mathcal{G}$ we have $|G \cap B| < |G \cap B^*|$.

Let G be a group considered in some iteration of the last loop in Algorithm 6. As observed above, the blue deficiency of G is strictly positive.

The blue deficiency of the union of all groups in $\mathcal{G}'$ is 0 because we have only removed blocks from $\mathcal{G}'$ up to this point and, by definition, a block has blue deficiency 0. Thus, there must be some other group $G' \in \mathcal{G}'$ with strictly negative blue deficiency. It cannot be that G' is bad, otherwise it has nonnegative blue deficiency. It also cannot be that G' is balanced or that it is a good group with a red representative, because such blocks also have nonnegative blue deficiency.

Therefore, G' must be good with a blue representative. Good blocks with blue representatives have blue deficiency exactly -1. Add this G' to $\mathcal{G}_0$. Iterating this argument with $G' + \mathcal{G}_0$, we add good groups with blue representatives to $\mathcal{G}_0$ until $G' + \mathcal{G}_0$ is a block. The layout in Figure 3.2 in Section 3.4 depicts a block constructed in this manner, the leftmost group in the figure is the bad group G and the remaining groups form $\mathcal{G}_0$.

After the last loop there are no groups with bad representatives or balanced groups. Furthermore, all good groups must have the same colour of representative by the second loop. If there were any good group, the blue deficiency of the union of groups in $\mathcal{G}'$ would then be nonzero, so there cannot be any good groups left. That is, at the end of the last loop there are no more groups in $\mathcal{G}'$. $\square$

3.4 Multi-swap Analysis

In this section we analyze Algorithm 4. We take a closer look at the cost change incurred by any valid test swap that swaps at most constant pairs of facilities of each color in a similar fashion as the analysis in Section 3.2. Thus we complete the proof for Theorem 3.1.1 in this section.

3.4.1 The Good, The Bad, The Randomized

Recall the example we gave previously in Figure 3.1. Client j with possibility of having both s_j, o_j closed within any single test swap are forced to reroute to $cent(s_j)$ and thus we have to use Lemma 3.2.2 to bound their cost. This in turns requires another test swap to cancel out the positive c_j terms. Intuitively we identify these clients as bad and the rest of the clients are good. Bad clients are more likely to incur additional test swaps and thus making bounding their cost harder. The workaround, similar as before, we will use a randomized argument to estimate the expected cost change for these clients. We will also show with an averaging argument we can bound the cost of a locally optimum solution nicely.

3.4.2 Four Cases

Recall that we are assuming $S = R \cup B$ is a locally optimum solution returned by Algorithm 4 and $O = R^* \cup B^*$ is some globally optimum solution. We assume $p = t^2 + 1$ for some sufficiently large integer t.

Focus on a single block T. For brevity, let $T^*_R = T \cap R^*$ and $T^*_B = T \cap B^*$ denote the red and blue facilities from the optimum solution in T. Similarly let $T_R = T \cap R$ and $T_B = T \cap B$ denote the red and blue facilities from the local optimum solution in T. The main goal of this section is to demonstrate the following inequality for block T.

Theorem 3.4.1. *For some absolute constant γ that is independent of t, we have*

$$0 \leq \sum_{j \in N^*(T^*_R \cup T^*_B)} \left[\left(1 + \frac{\gamma}{t}\right) c^*_j - c_j\right] + \sum_{j \in N(T_R \cup T_B)} \left[\left(4 + \frac{\gamma}{t}\right) \cdot c^*_j + \frac{\gamma}{t} \cdot c_j\right].$$

To prove Theorem 3.1.1, we simply consider summing the inequalities for all clients over all blocks. Because Theorem 3.4.1 applies to all blocks, We can sum the inequality from Theorem 3.4.1 for every block T and get:

$$0 \leq \sum_{j \in N^*(O)} \left[\left(1 + \frac{\gamma}{t}\right) c^*_j - c_j\right] + \sum_{j \in N(S)} \left[\left(4 + \frac{\gamma}{t}\right) \cdot c^*_j + \frac{\gamma}{t} \cdot c_j\right].$$

Now since each client j appears in both $N(S)$ and $N^*(O)$, then we can combine the two sums and get:

$$0 \leq \sum_{j \in \mathcal{C}} \left(5c^*_j - c_j + \frac{\gamma}{t}(2c^*_j + c_j)\right)$$

where γ is some absolute constant. Moving the c_j terms to the left and collect the coefficient, we get:

$$\sum_{j \in \mathcal{C}} c_j \leq \sum_{j \in \mathcal{C}} \left(\frac{5 + \frac{2\gamma}{t}}{1 - \frac{\gamma}{t}} c^*_j\right).$$

Recall $t \sim \sqrt{p}$ and Theorem 3.1.1 follows.

The analysis breaks into a number of cases based on whether T^*_R and/or T^*_B are large. In each of the cases, we use the following notation and assumptions. Let $\hat{i}$ denote the leader in T. Without loss of generality, assume all other $i \in T_B \cup T_R$ with $\deg(i) > 0$ are blue facilities. Let $\overline{B} = \{i \in T_B - \hat{i} : \deg(i) > 0\}$ and then $\mathrm{cent}(\overline{B})$ denotes $\{i^* \in T^*_B \cup T^*_R : \mathrm{cent}(i) = i^*, i \in \overline{B}\}$. Figure 3.2 illustrates this notation.

The swaps we consider in these cases are quite varied, but we always ensure we swap in cent(i) whenever some $i \in S \cap T$ with deg(i) > 0 is swapped out. This way, we can always bound the reassignment cost of each client j by using either Lemma 3.2.1 or Lemma 3.2.2.

3.4.3 Case $|T_R^*| \leq t^2, |T_B^*| \leq t$

In this case, we simply swap out all of $T_R \cup T_B$ and swap in all of $T_R^* \cup T_B^*$. Because $R \cup B$ is a locally optimum solution and because this swaps at most t^2 facilities of each colour, we have

$$0 \leq \text{cost}(S \cup (T_R^* \cup T_B^*) - (T_R \cup T_B)) - \text{cost}(S).$$

Of course, after the swap each client will move to its nearest open facility. Similar to what we did in the analysis for the simple example in Section 3.2, we explicitly describe a (possibly suboptimal) reassignment of clients to facilities to upper bound this cost change.

Each $j \in N^*(T_R^* \cup T_B^*)$ is moved from s_j to o_j which incurs an assignment cost change of exactly $c_j^* - c_j$. Each $j \in N(T_R \cup T_B) - N^*(T_R^* \cup T_B^*)$ is moved to $\phi(o_j)$. Note that $\phi(o_j) \notin T$ because $o_j \notin T_R^* \cup T_B^*$, so $\phi(o_j)$ remains open after the swap. By Lemma 3.2.1, the assignment cost change is bounded by $2c_j^*$. Every other client j that has not already been reassigned remains at s_j and incurs no assignment cost change. Thus,

$$0 \leq \sum_{j \in N^*(T_R^* \cup T_B^*)} (c_j^* - c_j) + \sum_{j \in N(T_R \cup T_B)} 2c_j^*$$

which is even better than what we are required to show for Theorem 3.4.1.

We note that the analysis Section 3.4.6 could subsume this analysis (with a worse constant), but we have included it here to serve as a simple warm up to the full analysis.

3.4.4 Case $|T_R^*| \geq t^2 + 1, |T_B^*| \geq t + 1$

We start by briefly discussing some challenges in this case. In the worst case, all of the $i_b \in T_B$ have deg(i) being very large. The issue here is that we need

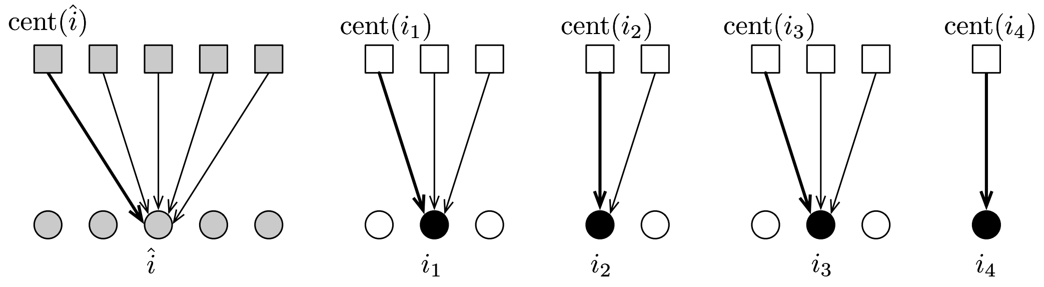

Figure 3.2: Illustration of a block T. The facilities on the top are in $T \cap O$ and the facilities on the bottom are in $T \cap S$. The directed edges depict ϕ, and the thick edges connect cent(i) to i. The facilities coloured black lie in $\mathcal{B}$, the facilities coloured white lie in $\mathcal{R}$, and the facilities coloured grey could either lie in $\mathcal{B}$ or $\mathcal{R}$. Note that $\overline{B} = \{i_1, i_2, i_3, i_4\}$ and $\text{cent}(\overline{B}) = \{\text{cent}(i_1), \text{cent}(i_2), \text{cent}(i_3), \text{cent}(i_4)\}$. The layout of the figure is suggestive of how the block was constructed by adding "good" groups to the initial bad group in the procedure of generating blocks.

to swap in each $i^*_b \in T^*_B$ in order to generate terms of the form $c^*_j - c_j$ for j with $o_j = i^*_b$. But this requires us to swap out some i_b. Since we do not have enough swaps to simply swap in all of $\phi^{-1}(i_b)$, we simply swap in $\text{cent}(i_b)$.

Any client j with s_j being closed and $o_j \in \phi^{-1}(i_b) - \text{cent}(i_b)$ cannot be reassigned to $\phi(o_j)$, so we send it to $\text{cent}(\phi(o_j))$ and use Lemma 3.2.2 to bound the reassignment cost. This leaves a term of the form $+c_j$, so we have to consider additional swaps involving $-c_j$ to cancel this out. These additional swaps cause us to lose a factor of roughly 5 instead of 3.

Another smaller challenge is that we do not want to swap out the leader $\hat{i} \in T \cap S$ for a variety of technical reasons. However, since $|T^*_R|$ and $|T^*_B|$ are both big, this is not a problem. When we swap in some $i^* \in T \cap O$, we will just swap out a randomly chosen facility in $T \cap S - \hat{i}$ of the same colour. The probability any particular facility is swapped in this way is very small. Ultimately, each facility in $T \cap S$ will be swapped out $2 + O(1/t)$ times in expectation.

To be precise, we partition the set of clients in $N(T_R \cup T_B)$ into two groups:

$$C_{bad} := N(\overline{B}) \cap N^*(T^*_R - \text{cent}(\overline{B})) \quad \text{and} \quad C_{ok} := N(T_R \cup T_B - \hat{i}) - C_{bad}.$$

Note that $N(\hat{i})$ is not included in C_{ok} because we will never close $\hat{i}$ in this case.

The first group is dubbed *bad* because there may be a swap where both s_j and $\phi(o_j)$ are closed yet o_j is not opened so we can only use Lemma 3.2.2 to bound their reassignment cost. In fact, some clients $j \in C_{ok}$ may also be involved in such a swap, but we are able to use an averaging argument for these clients to show that the resulting $+c_j$ term from using Lemma 3.2.2 appears with negligible weight and does not need to be cancelled.

We consider the following two types of swaps to generate our initial inequality.

- For each $i_b^* \in T_B^*$, choose a random $i_b \in T_B - \hat{i}$. If $i_b \notin \overline{B}$ (i.e. $\deg(i_b) = 0$) then simply swap out i_b and swap in i_b^*. If $i_b \in \overline{B}$ then swap out i_b and a random $i_r \in T_R - \hat{i}$ and swap in i_b^* and $\text{cent}(i_b)$.

- For each $i_r^* \in T_R^* - \text{cent}(\overline{B})$, swap in i_r^* and swap out a randomly chosen $i_r \in T_R - \hat{i}$.

By choosing facilities at "random", we mean uniformly at random from the given set and this should be done independently for each invokation of the swap.

Lemma 3.4.2.

$$0 \le \sum_{j \in N^*(T_B^* \cup T_R^*)} \left(\frac{t+1}{t} \cdot c_j^* - c_j \right) + \sum_{j \in C_{ok}} \left[\left(2 + \frac{5}{t} \right) c_j^* + \frac{1}{t} c_j \right] + \frac{t+1}{t} \sum_{j \in C_{bad}} (3c_j^* + c_j).$$

Proof. For brevity, we will let $\beta_R = \frac{|T_R|}{|T_R - \hat{i}|}$ and $\beta_B = \frac{|T_B|}{|T_B - \hat{i}|}$. Note that $\beta_R, \beta_B \le \frac{t+1}{t}$ and that either $\beta_R = 1$ or $\beta_B = 1$.

First consider a swap of the first type. This type of swap involves two situation depending on if the randomly chosen $i_b \in T_B - \hat{i}$ has $\deg(i_b) = 0$ or not. If $\deg(i_b) > 0$, we swap in $\{i_b^*, \text{cent}(i_b)\}$ and swaps out $\{i_b, i_r\}$. Because $R \cup B$ is a local optimum the cost of the solution does not decrease after performing this swap. We provide an upper bound on the reassignment cost.

Each $j \in N^*(\{i_b^*, \text{cent}(i_b)\})$ is reassigned from s_j to o_j and incurs an assignment cost change of $c_j^* - c_j$. Every client $j \in N(\{i_b, i_r\})$ that has not yet been reassigned is first moved to $\phi(o_j)$. If this $\phi(o_j)$ remains open, assign j

to it. By Lemma 3.2.1, the assignment cost for j increases by at most $2c_j^*$. If $\phi(o_j)$ is not open then $\phi(o_j) = i_b$ (because $\deg(i_r) = 0$) so we instead move j to $\text{cent}(\phi(o_j)) = \text{cent}(i_b)$. Lemma 3.2.2 shows the assignment cost increases by at most $3c_j^* + c_j$. This can only happen if $s_j \in \{i_r, i_b\}$ and $\phi(o_j) = i_b$.

Combining these observations and using slight overestimates, we see

$$0 \leq \sum_{j \in N^*(\{i_b^*, \text{cent}(i_b)\})} (c_j^* - c_j) + \sum_{\substack{j \in N(\{i_b, i_r\}) \\ \phi(o_j) \neq i_b}} 2c_j^* + \sum_{\substack{j \in N(\{i_b, i_r\}) \\ \phi(o_j) = i_b}} (3c_j^* + c_j). \qquad (3.1)$$

Now, if the random choice for i_b in the swap has $\deg(i_b) = 0$, then swapping $\{i_b\}$ out and $\{i_b^*\}$ in generates an even simpler inequality:

$$0 \leq \sum_{j \in N^*(i_b^*)} (c_j^* - c_j) + \sum_{j \in N(i_b)} 2c_j^*. \qquad (3.2)$$

To see this, just reassign each $j \in N^*(i_b^*)$ from s_j to o_j and reassign the remaining $j \in N(i_b)$ from s_j to $\phi(o_j)$ (which remains open because $\deg(i_b) = 0$) and use Lemma 3.2.1.

Consider the expected inequality that is generated for this fixed i_b^*. We start with some useful facts that follow straight from the structure of the block T and the swap we just performed.

- Any $j \in N^*(\text{cent}(\overline{B}))$ has o_j being opened with probability $\frac{1}{|T_B - \hat{\imath}|}$.
- Any $j \in C_{bad}$ has s_j being closed with probability $\frac{1}{|T_B - \hat{\imath}|}$.
- Any $j \in C_{ok} - N(T_R)$ has s_j being closed with probability $\frac{1}{|T_B - \hat{\imath}|}$. When this happens, if o_j is not opened then $\phi(o_j)$ must be open.

  That is, $s_j \in C_{ok}$ means $o_j \in T_B^* \cup \text{cent}(\overline{B})$. If $o_j \in T_B^*$ then $\phi(o_j) = \hat{\imath}$ (by the structure of block T) which remains open. If $o_j \in \text{cent}(\overline{B})$ then either $\phi(o_j)$ was not closed, or else $\text{cent}(\phi(o_j)) = o_j$ was opened.

- Any $j \in C_{ok} \cap N(T_R)$ has s_j being closed with probability $\frac{|\overline{B}|}{|T_B - \hat{\imath}|} \cdot \frac{1}{|T_R - \hat{\imath}|}$. If o_j and $\phi(o_j)$ are closed, then we move j to $\text{cent}(\phi(o_j))$. However, this can only happen with probability $\frac{1}{|T_B - \hat{\imath}|} \cdot \frac{1}{|T_R - \hat{\imath}|}$ since it must be that $\phi(o_j)$ is the blue facility that was randomly chosen to be closed.

Averaging (3.1) over all random choices and using some slight overestimates we see

$$\begin{aligned} 0 \ \leq \ & \sum_{j \in N^*(i^*_r)} (c^*_j - c_j) + \frac{1}{|T_B - \hat{i}|} \cdot \sum_{j \in N^*(\mathrm{cent}(\overline{B}))} (c^*_j - c_j) \\ & + \frac{1}{|T_B - \hat{i}|} \left[\sum_{j \in C_{bad}} (3c^*_j + c_j) + \sum_{j \in C_{ok} - N(T_R)} 2c^*_j \right] \\ & + \frac{1}{|T_B - \hat{i}|} \cdot \frac{1}{|T_R - \hat{i}|} \sum_{j \in C_{ok} \cap N(T_R)} (|\overline{B}| 2c^*_j + 3c^*_j + c_j). \end{aligned}$$

Summing over all $j \in N^*(T^*_B)$ shows

$$\begin{aligned} 0 \ \leq \ & \sum_{j \in N^*(T^*_B)} (c^*_j - c_j) + \beta_B \cdot \sum_{j \in N^*(\mathrm{cent}(\overline{B}))} (c^*_j - c_j) \\ & + \beta_B \cdot \left[\sum_{j \in C_{bad}} (3c^*_j + c_j) + \sum_{j \in C_{ok} - N(T_R)} 2c^*_j \right] \\ & + \frac{\beta_B}{|T_R - \hat{i}|} \cdot \sum_{j \in C_{ok} \cap N(T_R)} ((2|\overline{B}| + 3)c^*_j + c_j). \end{aligned} \tag{3.3}$$

Next, consider the second type of swap that swaps in some $i^*_r \in T^*_R - \mathrm{cent}(\overline{B})$ and swaps out some randomly chosen $i_r \in T_R - \hat{i}$. Over all such swaps, the expected number of times each $i_r \in T_R - \hat{i}$ is swapped out is $\frac{|T^*_R| - |\overline{B}|}{|T_R - \hat{i}|} = \beta_R - \frac{|\overline{B}|}{|T_R - \hat{i}|}$. In each such swap, we reassign $j \in N^*(i^*_r)$ from s_j to o_j and every other $j \in N(i_r)$ from j to $\phi(o_j)$ which is still open because $\deg(i_r) = 0$. Thus,

$$0 \leq \sum_{j \in N^*(T^*_R - \mathrm{cent}(\overline{B}))} (c^*_j - c_j) + \left(\beta_R - \frac{|\overline{B}|}{|T_R - \hat{i}|} \right) \cdot \sum_{j \in C_{ok} \cap N(T_R)} 2c^*_j$$

Scaling this bound by β_B, adding it to (3.3), and recalling $|T_R| \geq t^2$ shows

$$\begin{aligned} 0 \ \leq \ & \sum_{j \in N^*(T^*_B)} (c^*_j - c_j) + \beta_B \cdot \sum_{j \in N^*(T^*_R)} (c^*_j - c_j) \\ & + \beta_B \cdot \left[\sum_{j \in C_{bad}} (3c^*_j + c_j) + \sum_{j \in C_{ok} - N(T_R)} 2c^*_j \right] \\ & + \beta_B \cdot \sum_{j \in C_{ok} \cap N(T_R)} \left[\left(2\beta_R + \frac{3}{t^2} \right) \cdot c^*_j + \frac{1}{t^2} \cdot c_j \right]. \end{aligned}$$

Recall that $\beta_B, \beta_R \leq \frac{t+1}{t}$ and also $\beta_B \cdot \beta_R \leq \frac{t+1}{t}$ to complete the proof of Lemma 3.4.2. □

Our next step is to cancel terms of the form $+c_j$ in the bound from Lemma 3.4.2 for $j \in C_{bad}$. To do this, we again perform the second type of swap for each $i \in T^*_R - \text{cent}(\overline{B})$ but reassign clients a bit differently in the analysis.

Lemma 3.4.3.

$$0 \le \sum_{j \in C_{bad}} (c^*_j - c_j) + \frac{t+1}{t} \cdot \sum_{j \in C_{ok} \cap N(T_R)} 2c^*_j$$

Proof. For each $i^*_r \in T^*_R - \text{cent}(\overline{B})$, swap i^*_r in and swap out a randomly chosen $i_r \in T_r - \hat{i}$. Rather than reassigning all $j \in N^*(i^*_r)$ to i^*_r, we only reassign those in $C_{bad} \cap N^*(i^*_r)$. Since $\deg(i_r) = 0$ then any other $j \in N(i_r)$ must have $\phi(o_j) \neq i_r$ and can be reassigned to $\phi(o_j)$ and which increases the cost by at most $2c^*_j$ by applying Lemma 2.1.3.

Summing over all i^*_r, observing that $C_{bad} \subseteq T^*_R - \text{cent}(\overline{B})$, and also observing that each $j \in C_{ok}$ has s_j closed at most $\beta_R \le \frac{t+1}{t}$ times in expectation, we derive the inequality stated in Lemma 3.4.3. □

Adding the bounds stated in Lemmas 3.4.2 and 3.4.3 we get:

$$0 \le \sum_{j \in N^*(T^*_B \cup T^*_R)} \left(\frac{t+1}{t} \cdot c^*_j - c_j\right) + \sum_{j \in C_{ok}} \left[\left(4 + \frac{5}{t}\right) c^*_j + \frac{1}{t} c_j\right] + \sum_{j \in C_{bad}} \left[\left(4 + \frac{3}{t}\right) c^*_j + \frac{1}{t} c_j\right]. \quad (3.4)$$

Now note that $C_{ok} \cap C_{bad} = \emptyset$ and $C_{ok}, C_{bad} \subseteq N(T_B \cup T_R)$ because of the definitions of C_{ok} and C_{bad}. So we can combine the two sums and take the upper bound and get:

$$0 \le \sum_{j \in N^*(T^*_B \cup T^*_R)} \left(\frac{t+1}{t} \cdot c^*_j - c_j\right) + \sum_{j \in N(T_B \cup T_R)} \left[\left(4 + \frac{5}{t}\right) c^*_j + \frac{1}{t} c_j\right].$$

Setting $\gamma = 5$ shows that Theorem 3.4.1 holds in this case.

3.4.5 Case $|T^*_R| \ge t^2 + 1, |T^*_B| \le t$

In this case, we start by swapping in all of T^*_B and swapping out all of T_B (including, perhaps, $\hat{i}$ if it is blue). Let $m = |\text{cent}(T_B)| - 1$ if $\hat{i}$ is blue and cent($\hat{i}$) is also blue, or $m = |\mathrm{cent}(T_B)|$ otherwise. In the same swap, we also swap in cent(T_B) and swap out a random subset of size m of facilities in $T_R - \hat{i}$. This is possible as $|T_R - \hat{i}| \geq t \geq |\mathrm{cent}(T_B)| \geq m$. By random subset, we mean among all subsets of $T_r - \hat{i}$ of size m, choose one uniformly at random.

As with Section 3.4.4, we begin with a definition of bad clients that is specific to this case:

$$C_{bad} := N(T_B) \cap N^*(T_R^* - \mathrm{cent}(T_B)).$$

Clients $j \in C_{bad}$ may be involved in swaps where both s_j and $\phi(o_j)$ are closed yet o_j is not opened and we cannot make this negligible with an averaging argument.

Lemma 3.4.4.

$$0 \quad \leq \quad \sum_{j \in N^*(T_B^* \cup \mathrm{cent}(T_B))} (c_j^* - c_j) + \frac{1}{t} \sum_{j \in N(T_R)} (3c_j^* + c_j) + \sum_{j \in C_{bad}} (3c_j^* + c_j)$$

Proof. After the swap, reassign every $j \in N^*(T_B^* \cup \mathrm{cent}(T_B))$ from s_j to o_j, for a cost change of $c_j^* - c_j$. Every other j that has s_j being closed is first reassigned to $\phi(o_j)$. If this is not open, then further move j to cent(o_j) which must be open because the only facilities $i \in T_R \cup T_B$ with $\deg(i) > 0$ that were closed lie in T_B and we opened cent(T_B).

If $j \in N(T_B) - C_{bad}$ then $o_j \in T_B^* \cup \mathrm{cent}(T_B)$ and we have already assigned j to o_j. If $j \in C_{bad}$ then we have moved j to cent($\phi(o_j)$) and the cost change is $3c_j^* + c_j$ by Lemma 3.2.2.

Finally, if $j \in N(T_R)$ then we either move j to $\phi(o_j)$ or to cent($\phi(o_j)$) if $\phi(o_j)$ is not open. The worst-case bound on the reassignment cost is $3c_j^* + c_j$ by Lemmas 3.2.1 and 3.2.2. However, note that $s_j \in T_R$ is closed with probability only $1/t$, since we close a random subset of $T_r - \hat{i}$ of size at most t and $|T_r - \hat{i}| \geq t^2$.

□

To generate the desired inequality for block T, we require some additional swaps, described in the following: For each facility $i_r^* \in T_R^* - \mathrm{cent}(T_B)$, swap in i_r^* and swap out a randomly chosen $i_r \in T_R - \hat{i}$. The analysis of these swaps is essentially the nearly identical swaps in Section 3.4.4, so we omit it and merely summarize what we get by combining the resulting inequalities with the inequality from Lemma 3.4.4.

Lemma 3.4.5.

$$0 \leq \sum_{j \in N^*(T_R^* \cup T_B^*)} (c_j^* - c_j) + \sum_{j \in N(T_R)} \left(\frac{t^2+1}{t^2} \cdot 2c_j^* + \frac{1}{t} \cdot (3c_j^* + c_j) \right) + \sum_{j \in C_{bad}} (3c_j^* + c_j)$$

We cancel the $+c_j$ terms for $j \in C_{bad}$ with one further collection of swaps. For each $i_r^* \in T_R^* - \text{cent}(T_B)$ we swap in i_r^* and a randomly chosen $i_r \in T_R - \hat{i}$. The following lemma summarizes a bound we can obtain from these swaps. It is proven in essentially the same way as Lemma 3.4.3.

Lemma 3.4.6.

$$0 \leq \sum_{j \in C_{bad}} (c_j^* - c_j) + \frac{t^2+1}{t^2} \cdot \sum_{j \in N(T_R)} 2c_j^*.$$

Adding this to the bound from Lemma 3.4.5 shows

$$0 \leq \sum_{j \in N^*(T_R^* \cup T_B^*)} (c_j^* - c_j) + \sum_{j \in N(T_R \cup T_B)} \left(\frac{t^2+3t+1}{t^2} \cdot 4c_j^* + \frac{1}{t} \cdot c_j \right).$$

3.4.6 Case $|T_R^*| \leq t^2, |T_B^*| \geq t+1$

Because $\phi^{-1}(i) \subseteq T_R^*$ and $\deg(i) > 0$ for each $i \in \overline{B}$, then $|\overline{B}| \leq t^2$ as well. We will swap all of T_R^* for all of T_R, but we will also swap some blue facilities at the same time. Let $B' = \overline{B}$ and let $\overline{B}'$ be an arbitrary subset of T_B^* of size $|\overline{B}|$.

If $\hat{i} \notin T_R \cup B'$ then add $\hat{i}$ to B'. If $\text{cent}(\hat{i}) \notin T_R^* \cup \overline{B}'$ then add $\text{cent}(\hat{i})$ to $\overline{B}'$. At this point, $\left| |\overline{B}'| - |B'| \right| \leq 1$ Add an arbitrary $i_b^* \in T_B^* - \overline{B}'$ to $\overline{B}'$ or $i_b \in T_B - B'$ to B' to ensure $|\overline{B}'| = |B'|$.

We begin by swapping out $T_R \cup B'$ and swapping in $T_R^* \cup \overline{B}'$. The following list summarizes the important properties of this selection, the first point emphasizes that this swap will not improve the objective function since S is a locally optimum solution for the p-swap heuristic where $p = t^2 + 1$.

- $|B'| = |\overline{B}'| \leq t^2 + 1$ and $|T_R^*| \leq t^2$.

- T_R^* was swapped in and T_R was swapped out.
- For each $i \in T_R \cup T_B$ with $\deg(i) > 0$, i was swapped out and $\text{cent}(i)$ was swapped in.

The following is precisely the clients j that will be moved to $\text{cent}(\phi(o_j))$ in our analysis.

$$C_{bad} := [N(T_R \cup B') - N^*(T_R^* \cup \overline{B}')] \cap \{j : \phi(o_j) \in T_R \cup B'\}.$$

As before, define $C_{ok} = N(T_R \cup T_B) - C_{bad}$.

The following bound is generated from swapping out $T_R \cup B'$ and swapping in $T_R^* \cup \overline{B}'$ and follows from the same arguments we have been using throughout the paper.

Lemma 3.4.7.

$$0 \leq \sum_{j \in N^*(T_R^* \cup \overline{B}')} (c_j^* - c_j) + \sum_{j \in C_{ok} \cap N(T_R \cup B')} 2c_j^* + \sum_{j \in C_{bad}} (3c_j^* + c_j)$$

Next, let $\kappa_B : (T_B^* - \overline{B}') \to (T_B - B')$ be an arbitrary bijection of the remaining blue facilities that were not swapped. For every $i_b^* \in T_B^* - \overline{B}'$, consider the effect of swapping in i_b^* and swapping out $\kappa_B(i_b^*)$. Note that every facility i_b swapped out in this way has $\deg(i_b) = 0$. So we can derive two possible inequalities from such swaps.

$$0 \leq \sum_{j \in N^*(i_b^*)} (c_j^* - c_j) + \sum_{j \in N(\kappa_B(i_b^*))} 2c_j^* \quad \text{and} \quad 0 \leq \sum_{j \in N^*(i_b^*) \cap C_{bad}} (c_j^* - c_j) + \sum_{j \in N(\kappa_B(i_b^*))} 2c_j^*. \tag{3.5}$$

The second inequality follows from only reassigning clients $j \in N^*(i_b^*) \cap C_{bad}$ from s_j to o_j.

Adding the bound in Lemma 3.4.7 to the sum of both inequalities over all $i_b^* \in T_B^* - \overline{B}'$ and noting that $\kappa_B(T_B^* - \overline{B}) \cap (T_R \cup B') = \emptyset$, we see

$$0 \leq \sum_{j \in N^*(T_R^* \cup T_B^*)} (c_j^* - c_j) + \sum_{j \in N(T_R \cup T_B)} 4c_j^*.$$

Now we see all four cases we were able to get an inequality we desire. Theorem 3.4.1 simply follows by extending to a coarse upper bound on the coefficients over all clients.

We remind the reader that we only used at most $2(k_r + k_b)$ swaps in the analysis. This allows us to adapt the standard modification that appeared in the proof of Lemma 2.1.6 to ensure Algorithm 4 runs in polynomial time while only losing a $(1+\epsilon)$ multiplicative factor to the approximation guarantee. Intuitively, we consider to modify the natural local search algorithm for Budgeted Red-Blue Median in the same fashion as shown by Algorithm 3 with the difference being we put $\left(1 - \frac{\epsilon}{2(k_r+k_b)}\right)$ instead of $\left(1 - \frac{\epsilon}{k}\right)$. The rest of the proof should follow the proof of Theorem 2.1.6 closely so we omit the details of the proof here.

Chapter 4

Tight Gap Example for Local Search Algorithm

In this chapter we show that the analysis shown in Section 3.4 has a matching lower bound. By the definition of an α-approximation, in order to show that $5+\epsilon$ is the best possible approximation guarantee using the local search algorithm in Section 3.1 all that is required is to produce a worst case scenario where the locality gap matches the approximation factor.

Theorem 4.0.1. *For any integers p, ℓ with $1 \le p \le \ell/2$, there is an instance of* Budgeted Red-Blue Median *that has a locally-optimum solution for the p-swap heuristic with cost at least $\left(5+\frac{2}{p}-\frac{10p}{\ell+1}\right)\cdot OPT$.*

Theorem 4.0.1 provides a matching lower bound for the natural local search algorithm on Budgeted Red-Blue Median. Notably the theorem guarantees a $(7-\epsilon)$ matching lower bound for a single swap analysis by letting $p=1$.

In the following sections, we first present a simple set up of a bad gap example to illustrate the analysis for a single swap case. From there we will present a bad gap example for the general p-swap setting. And finally we extend this approach to apply it to the multi-colour setting.

4.1 Simple Single Swap Bad Gap Example

We will first present a simple case where we are only allowed single swap analysis, or more precisely, for each test swap we can swap up to one pair of

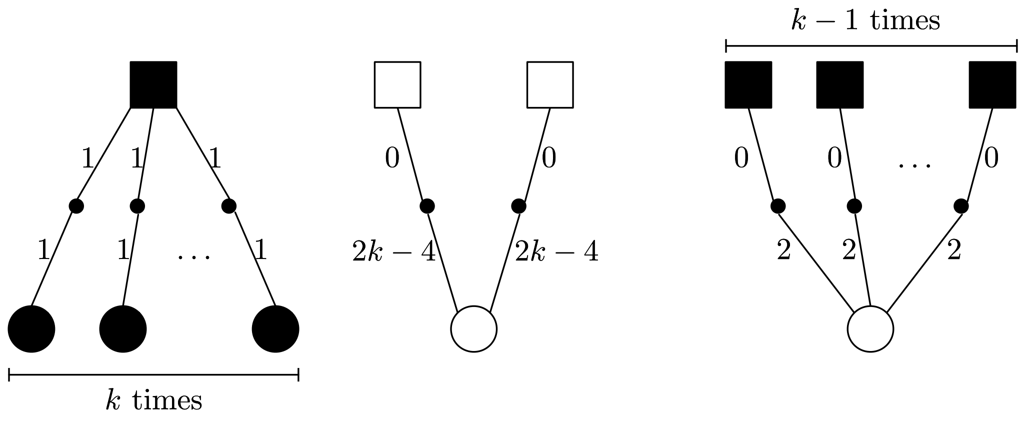

Figure 4.1: This is a illustration of an instance of Red Blue Median with a bad locality gap with single swap heuristics local search. We are given $|R| = |R^*| = 2$ and $|B| = |B^*| = k$ with squares depicts O and circles are facilities in S. We are also given the set of clients C drawn as smaller solid dots. The edges in the graph shows the distance between any two locations and the metric is shortest distance metric. Consider the three groups of facilities and clients in the figure, judging from the clustering we name them left group, middle group and the right group. We show in this instance the cost$(S) \geq (7 - \frac{10}{k})OPT$

blue facilities and/or one pair of red facilities. We show in this example the following:

Theorem 4.1.1. *Figure 4.1 depicts an instance of* Budgeted Red-Blue Median *such that* $\text{cost}(S) \geq (7 - \frac{10}{k})\text{cost}(O)$ *and S is locally optimum solution.*

Consider the instance depicted in figure 4.1. We are given the locally optimum solution S produced by the local search algorithm for the single swap heuristics and a global optimum solution O. Both S and O are consisting of k blue facilities and 2 red facilities. We divide the facilities into three groups as depicted in the figure, for the left group, for each blue facility i in S there exists a client such that $d(i, j) = d(i', j)$ where i' is the single facility in the OPT in the group. For the middle group we only have a single red facility $i \in S$ and two red facilities $i' \in O$, and for each i' in the middle group, we have one client j with $d(i, j) = 2k - 4$ and letting j to be infinitely close to i' so that the distance $d(j, i') = 0$. For the right group, we have $k - 1$ blue facilities in O and a single red facility in S. There also exists one client for each blue facilities at infinitely close distance and a distance of 2 away from the red facility as depicted in the figure.

Now we prove Theorem 4.1.1. And in order to do so, it is sufficient to show the following:

- S is indeed a locally optimum solution, i.e. there is no single test swap that finds a cheaper solution
- $\text{cost}(S) \geq (7 - \epsilon) \cdot \text{cost}(O)$

The second point is very easy to verify: Recall the cost of a solution is the sum of distance of all clients to their closest open facility. Therefore in S, each client j in the left group is assigned to the facility directly below it for any other facility i' the distance $d(i', j) \geq 3$ by the shortest path metric. And since there is only one single open facility in S for both the middle and right group, it's trivial to see that all the clients are assigned to that facility. Therefore summing up the distance and we get the cost of S:

$$\text{cost}(S) = k \cdot 1 + 2 \cdot (2k - 4) + 2 \cdot (k - 1) = 7k - 10$$

For O, following the obvious clients assignment, note the cost increase for assigning clients in the middle and right group is 0, we get the following:

$$\text{cost}(O) = k$$

And for a large k we see that the gap approaches 7 between S and O. Now all that remains is to show that S is indeed a locally optimum solution. Since we are only allowing a single pair of facility to be swapped, we can verify the first point by verifying all possible test swaps yields a cost change of non-negative value. Now we analyze all possible test swaps:

Only Swap A Single Pair Of Blue Facilities

Let $i \in B$ and $i \in B^*$, we consider the swap solution $(S - i + i^*)$. Note i must be from the left group. First suppose i^* is from the left group. Since all clients have a distance of 1 to any closest open facility, the cost of the left group is still k there is no cost change: $\text{cost}(S - i + i^*) - \text{cost}(S) = 0$. The only other option is to let i^* be from the right group. When we open i^*, the client j' that is nearest to i^* in the right group gets to reassign to i^* and for simplicity reasons, from now we will call such clients are associated with the facility. So the cost change of the client associated with i^* is exactly $0 - 2 = -2$. And for the client j associated with i from the left group, j must be reassign to another open facility and since no facility in O in the left group is open, the cost change for reassigning j is $3 - 1 = 2$. Thus the cost change overall reassignments is $\text{cost}(S - i + i^*) - \text{cost}(S) = 0$.

Only Swap A Single Pair Of Red Facilities

We cannot close any red facility from the right group because we are restricted not to open any blue facility in this case. Otherwise there won't be any open facility in the right group so the cost of assigning any client from the right group would be unbounded. So the only possible test swap in this case is to swap one pair of red facilities in the middle group. In this case, the cost change is clearly $(2k - 4) - (2k - 4) = 0$.

Swap Both Blue And Red Facilities

It is obvious to see that if the pair of red facilities we chose are both from the middle group, then we can simply combine the analysis for the previous two cases and see that the cost change is 0. Therefore the only case we have not considered is having $r^* \in R^*$ from the middle group and $r \in R$ from the right group. Now since we closed r we must open another facility in the right group, so let $b^* \in B^*$ from the right group and this forces us to choose $b \in B$ from the left group. The test swap we consider is $S - r - b + r^* + b^*$. For the left group, we closed b so the client associated with b has to travel a distance of 3 and has a cost change of $3 - 1 = 2$. For the middle group, we opened r^*, the cost change of the client associated with r^* is $0 - (2k - 4) = -(2k - 4)$. For the right group, we closed r so now all the clients has to be rerouted to the only open facility b^*. The client associated with b^* saves a cost of $0 - 2 = -2$ but every other client j now has to travel to b^*. Note that $d(j, r) = 2$ and

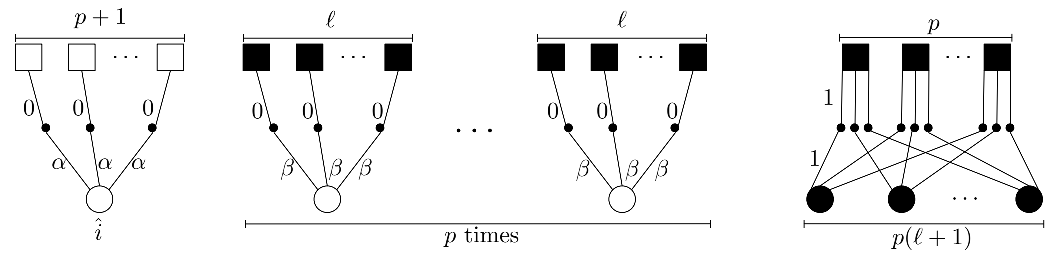

Figure 4.2: Illustration of the bad locality gap. Blue facilities are depicted with black and red facilities are depicted with white. The top facilities are the global optimum and the bottom are the local optimum (all of $\mathcal{R}$ and $\mathcal{B}$ is depicted in the picture). Each client is represented by a small black dot. The metric is the shortest path metric of the presented graph, if two locations are not connected in the picture then their distance is a very large value. Every edge in the right-most group with $p^2(\ell+1)$ clients has length 1. Recall $\beta = 2p$ and $\alpha = (\ell - p)2p$.

$d(j, b^*) = d(j, r) + d(r, b^*) = 4$ so the cost change for each client j is $4 - 2 = 2$ and there are $k - 2$ such clients. Therefore adding up all the numbers we get the cost change for the right group is $2(k-2) - 2$. So combine all the groups together we get $\text{cost}(S - r - b + r^* + b^*) - \text{cost}(S) = 2 - (2k-4) + 2(k-2) - 2 = 0$.

Summary

Now we have shown that any test swap allowing up to a single swap yields a non-negative cost change value so S is indeed a locally optimum solution. Theorem 4.1.1 follows from there.

In the next section we will extends the analysis for the case of multi-swap heuristics. The analysis again follows the same fashion of the single swapped case however the added parameters increases the complexity of presenting our results, but the idea generally follows through.

4.2 Multi-Swap Instance Description and Cost Gap

Now we prove Theorem 4.0.1. Let p, ℓ be integers satisfying $p \geq 1$ and $\ell \geq 2p$. Consider the instance with $k_r = p + 1$ and $k_b = p(\ell + 1)$ depicted in Figure 4.2. Here, $\beta = 2p$ and $\alpha = \beta \cdot (\ell - p)$.

The cost of the local optimum solution is $\alpha \cdot (p+1) + \beta \cdot p\ell + p^2(\ell+1)$ and the cost of the global optimum solution is simply $p^2(\ell+1)$. Through some careful simplification, we see the local optimum solution has cost at least $5 + \frac{2}{p} - \frac{10p}{\ell+1}$ times the global optimum solution.

4.3 Locality Optimality

We verify that the solution depicted in Figure 4.2 is indeed a locally optimum solution. Suppose $0 \leq R \leq p$ red facilities and $0 \leq B \leq p$ blue facilities are swapped. We break the analysis into four simple cases.

In what follows, we refer to the leftmost collection of only red facilities in Figure 4.2 as the *left group*, the rightmost collection of only blue facilities as the *right group*, and the remaining facilities as the *middle group*. We also let the term *subgroup* refer to one of the p smaller collections of facilities in the middle group. In each case, let $B' \leq B$ denote the number of global optimum facilities from the middle group that are swapped in. Recall that $\hat{i}$ denotes the local optimum facility in the left group.

Case $R = 0$

The only clients that can move to a closer facility are the B' clients in the middle group that have their associated optimum facilities swapped in. Also, precisely $B \cdot (p - B + B')$ clients in the right group are not adjacent to any open facility so their assignment cost increases by 2.

Overall, the assignment cost change is exactly $2B \cdot (p - B + B') - \beta B'$. As $\beta = 2p$ and $B \leq p$, this quantity is minimized at $B' = B$ leaving us with a cost change of $2Bp - \beta B = 0$. So, if $R = 0$ then no choice of blue facilities leads to an improving swap.

Case $R \geq 1$ and $\hat{i}$ is not swapped out.

In the left group, precisely R clients move to their close facility and the total savings is $-\alpha R$. In the middle group, precisely B' clients move to their close facility and the total savings is $-\beta B'$.

In fact, it is easy to see that the cheapest such swap occurs when the $B' \leq p \leq \ell$ facilities in the middle group that are swapped in are part of subgroups where the local optimum facility is swapped out (which is why we assume $R \geq 1$). So, there are exactly $R\ell - B'$ other clients j where both o_j and s_j are closed and each pays an additional $\geq \beta$ to be connected. Finally, the right group pays an additional $2B(p - B + B')$ to be connected.

Overall, the cost increases by $2B(p - B + B') + \beta(R\ell - 2B') - \alpha R$. As $2B - 2\beta = 2B - 4p \leq -2p$, this is minimized at $B' = B$. The cost change is then $2Bp + \beta(R\ell - 2B) - \alpha R$. Recall that $\alpha = (\ell - p)\beta < \ell\beta$, so this is, in turn, minimized when $R = 1$.

Reducing further, the cost change is $2Bp + \beta\ell - 2B\beta - \alpha$. Setting $B = p$ to maximize, the change is $2p^2 + 2p\ell - 4p^2 - (\ell - p)2p = 0$. So, no swap that swaps at least one red facility but not $\hat{i}$ can find a cheaper solution.

Case $R = 1$ and $\hat{i}$ is swapped out.

The cost change in the left group is $(p - 1)\alpha \geq 0$ since p clients must move an additional α and only one client saves α. The cost change from the remaining groups is the same as in the first case $R = 0$, so the overall assignment cost does not decrease.

Case $R \geq 2$ and $\hat{i}$ is swapped out.

The cost change in the first group is exactly $(p + 1 - 2R)\alpha$. Similar to the second case, the cost change in this case is minimized when each subgroup that has its local optimum facility closed also has one of its global optimum facility opened, and all B' facilities opened in the middle group belong to a subgroup having its local optimum closed.

The cost change is then $2B(p-B+B')+\beta((R-1)\ell-2B')+\alpha(p+1-2R)$. Again, this is minimized at $B' = B$ which yields a cost change of $2Bp+\beta((R-1)\ell - 2B) + \alpha(p + 1 - 2R)$. Now, $\beta\ell \leq 2\alpha$ because $\ell \geq 2p$, so this is further minimized at $R = p$ and the cost increase is $2Bp+\beta((p-1)\ell-2B)-\alpha(p-1)$. Again, setting $B = p$ to minimize the cost change we see it is $2p^2 + \beta((p - 1)\ell - 2p) - \alpha(p - 1)$.

Expanding with $\beta = 2p$ and $\alpha = 2p(\ell - p)$, the cost change finally seen to be

$$-2p^2 + 2p(p-1)\ell - 2p(\ell - p)(p-1) = 2p^3 - 4p^2.$$

The last expression is nonnegative for $p \geq 2$.

4.3.1 Summarizing

No matter which $\leq p$ red and $\leq p$ blue facilities are swapped, the above analysis shows the assignment cost does not decrease. The only potentially concerning aspect is that the very last case derived an inequality that only holds when $p \geq 2$. Still, this analysis does apply to the single-swap case (i.e. $p = 1$) since the last case with $R \geq 2$ does not need to be considered when $p = 1$.

4.4 Extending to t Colors

A natural extension is to apply this approach for a fixed constant types of facilities, we name the following problem setting the $t-$COLOR FACILITY MEDIAN(TCFM). We are given a set of facilities $\mathcal{F}$ and a set of clients $\mathcal{C}$. Given a constant t of colors and a mapping color(i) : $\mathcal{B} \cup \mathcal{R} \rightarrow \{1, 2, 3, ..., \mathrm{t}\}$ that maps each facility to one color. We are given constraints to open exactly k_i of each $1 \leq i \leq t$ colored facilities. In this setting, the goal is to minimize the total service cost of all clients.

We consider the following natural local search algorithm for TCFM:

Algorithm 7 The p-Swap Heuristic for the TCFM

```
Let $S$ be an arbitrary feasible solution and **constant** $P$.
**while** there is some feasible solution $S'$ with **the difference in the number of facilities at most $P$ for each individual color** and cost($S'$) < cost($S$) **do**
    $S \leftarrow S'$
**end while**
**return** $S$
```

Note that we simply extended Algorithm 4 by adapting the constraints to t colors to obtain Algorithm 7. Algorithm 7 outputs a locally optimum solution for our problem setting and in this section we prove:

Theorem 4.4.1. *For any $t > 2$, there exists an instance of TCFM for some P, L such that*

$$\text{cost}(S) \geq \left(2t + 1 + \frac{2}{P} - \epsilon(P, L)\right) \text{cost}(O)$$

where O is a global optimum solution and S is an locally optimum solution to the $p-$swap heuristic and $\epsilon(P, L) \to 0$ as $L \to \infty$ with P fixed.

Proof. Consider the instance of TCFM in Figure 4.3. Let O be the global optimum solution depicted as squares in Figure 4.3 and S be the locally optimum solution drawn as circles and we let $S \cap O = \emptyset$. Let U_i denote the set of facilities of color i for $1 \leq i \leq t$ and let $|O \cap U_i| = |S \cap U_i| = k_i$.

We partition the instance into groups $G_0, G_1, G_2, ..., G_t$ as depicted in Figure 4.3, a total of $t + 1$ groups. Now let $O_i = O \cap G_i$ and $S_i = S \cap G_i$ for all $0 \leq i \leq t$ and we let the distance $d(u, v)$ to be infinite for any two nodes u, v that is not from the same group.

For group G_0, we have $O_0 \cap O \subseteq U_1$ and $\hat{i} \in U_1$. We have $P + 1$ facilities in O_0 and there is one client at same location for each facility in O_0 and the distance between each client and $\hat{i}$ is c_0. For each group G_i from $1 \leq i \leq t-1$, consider a *part* containing L facilities in O of color $i+1$ and one facility from S of color i. There are L clients in each part and located as shown in Figure 4.3 and there in total PL^{i-1} identical parts that forms group G_i. Finally we have $|O_t| = P$ and $|S_t| = PL^{t-1} + P$ with $O_t, S_t \subseteq U_t$. For each pair of facilities $(u, v), u \in O_t, v \in S_t$ we have one client that is located exactly distance of 1 away from u, v.

Now we check the cost of S. For G_0 we have $\text{cost}(S_0) = 2P(P+1)(L-P)^{t-1}$ and $\text{cost}(O_0) = 0$. For each group G_i except G_0, G_t, we have $\text{cost}(S_i) = 2P^2(L-P)^{t-i-1}L^i$ and $\text{cost}(O_i) = 0$. Lastly we have group G_t with $\text{cost}(S_t) = \text{cost}(O_t) = P^2L^t + P^2$.And sum over all groups we have:

$$\begin{aligned}
\text{cost}(S) &= \sum_{i=0}^{t-1} 2P^2(L-P)^{t-i-1}L^i + P^2L^{t-1} + P^2 + 2P(L-P)^{t-1} \\
\text{cost}(O) &= P^2L^{t-1} + P^2
\end{aligned}$$

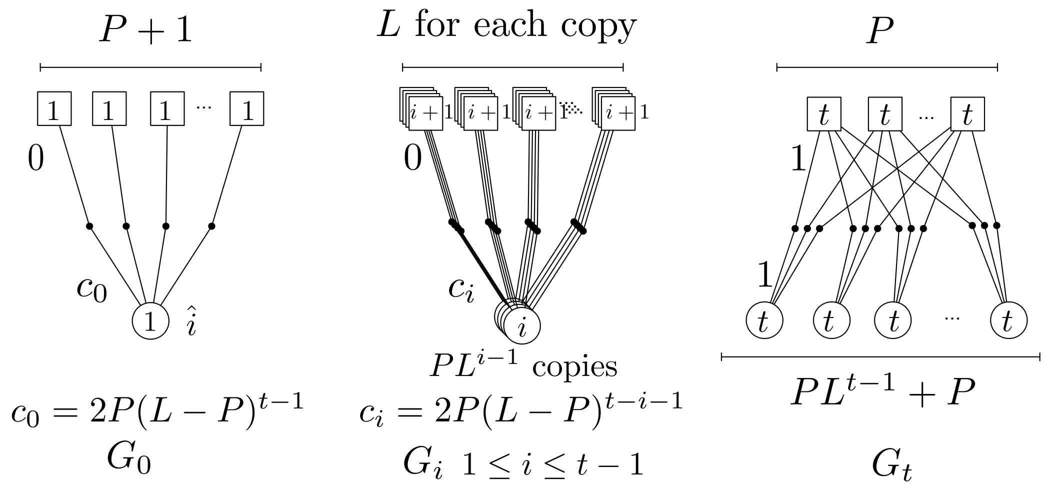

Figure 4.3: A TCFM instance with locality gap of $2t+1$. The squares represent O and circles for S and we let $S \cap O = \emptyset$. The instance is partitioned into groups: $G_0, G_1, G_2, ..., G_t$ as depicted in the figure. For this instance we will require $1 \le P \le L \le PL^{t-1} + P \le k$ where P, L, k are positive integers and we also require L to be significantly greater than P.

It's clear to see that when L is significantly larger than P we have $\text{cost}(S) \sim 2tP^2L^{t-1} + P^2L^{t-1} + P^2 + 2PL^{t-1}$ and $\text{cost}(O) = P^2L^{t-1} + P^2$. This shows:

$$\frac{\text{cost}(S)}{\text{cost}(O)} = 2t + 1 + \frac{2}{P} - \epsilon(P, L)$$

Next we prove that S is indeed a locally optimum solution. To do this, we need to show any valid test swaps have a non negative value in the cost change. We first define some notation to describe test swaps. Consider a solution S' obtained by executing a valid test swap on S. Let $v_i = |S' \cap O_i|$ for each $0 \le i \le t$.

Now consider following lemmas:

Lemma 4.4.2. *Consider solution S', we have $\text{cost}(S') \ge \text{cost}(S)$ if any of the following holds:*

1 $v_0 = 0$ and $\hat{i}$ is closed

2 $v_0 > v_1 + 1$ and $\hat{i}$ is closed

3 $v_0 > v_1$ and $\hat{i}$ is not closed

4 $v_i > v_{i+1}$ *for* $1 \leq i \leq t-2$

Proof. For item 1, since $\hat{i} \notin S'$, then $G_0 \cap S' = \emptyset$, all clients in G_0 have infinite service cost by the definition of the distance function.

For item 2, we have $v_0 \geq v_1 + 2$. There is $v_1 + 1$ facilities in S_1 gets closed in order to balance the color. However we only opened v_1 facilities in O_1 therefore there must be at least one part of the group has no open facility. And by the definition of the distance function, the clients in that part have infinite service cost. The proof for item 3 follows similarly.

For item 4, we consider groups G_i for $1 \leq i \leq t-2$. We have $v_i \geq v_{i+1} + 1$. In order to balance the color in the test swap, it must be that v_i facilities are closed and v_{i+1} opened in G_{i+1}. And since v_i strictly greater than v_{i+1} then there must exists a part in G_{i+1} contains no open facility which in turn contributes infinitely to the overall service cost.

□

Lemma 4.4.3. *Consider solution S', there exists a solution S'' having*

$$\text{cost}(S'') \leq \text{cost}(S')$$

if any of the following holds:

1 $v_0 = 1$ *and* $\hat{i}$ *is closed*

2 $v_t > 0$

Proof. For item 1, consider a solution S'' where we simply do not open or close any facilities in G_0 and $S'' \cap G_i = S' \cap G_i$ for $1 \leq i \leq t$ which implies cost contribution for clients that are not in G_0 is the same for both solutions S, S''. Now S' opened one facility in O_0 and closed $\hat{i}$, therefore all clients in G_0 has to be reassigned. We have a cost contribution of $2Pc_0$ for S'' and a cost contribution of $(P+1)c_0$ for S. Now we know $c_0 > 0$ and $P \geq 1$ so we conclude that $\text{cost}(S'') \leq \text{cost}(S')$.

Next for item 2, consider a solution S'' where we do not open any facilities in O_t but instead open additional v_t facilities in O_{t-1}. Again there is no difference in groups $G_0, G_1 ... G_{t-2}$ compare to S so we focus on comparing the cost contribution in $G_{t-1} \cup G_t$. Consider the cost difference between the two solution: In S'' we open additional v_t facilities in O_{t-1} thus saving a cost of $v_t c_{t-1}$; We opened 0 facilities in O_t so there is $P(v_t + v_{t-1})$ clients are distance 3 away from an open facility. However in S' there are $(P - v_t)(v_t + v_{t-1})$ clients needs to travel a distance of 3 to an open facility, comparing the cost of the two solutions we have the following:

$$\text{cost}\,(S') - \text{cost}(S'') = v_t c_{t-1} - 2v_t(v_t + v_{t-1})$$

Now $c_{t-1} = 2P$ and $v_t + v_{t-1} \le P$ because v_t, v_{t-1} opens the same colored facilities, we have $\text{cost}(S') - \text{cost}(S'') \ge 0$. □

Lemma 4.4.2 and Lemma 4.4.3 provides us a way to rule out some test swaps that we don't need to consider. And we conclude that we only need to consider test swap T satisfying the following conditions:

- $\hat{i}$ is not closed: $0 \le v_0 \le v_1 \le \ldots \le v_{t-1} \le P$ and $v_t = 0$.
- $\hat{i}$ is closed : $2 \le v_0 \le v_1 + 1$ and $1 \le v_1 \le \ldots \le v_{t-1} \le P$ and $v_t = 0$.

So from now on we only consider solution S' that satisfy the above conditions:

Case: $\hat{i}$ is not closed

Suppose a is the smallest index where $v_a > 0$ and $0 \le a \le t-1$, we open v_a facilities in O_a so we get cost change $-c_a v_a$. Then for each subsequent group G_i, we open v_i facilities in O_i, thus each of $v_{i-1}L - v_i$ clients needs to be rerouted and pay an extra c_i. The cost change is $+c_i(v_{i-1}L - v_i) - c_i v_i$ for each group G_i. Finally we close exactly v_{t-1} facilities in S_t and get cost change $+2Pv_{t-1}$ as any clients that incidents to no open facilities have to travel a extra distance of 2 to be assigned at an open facility. And to summarize:

$$\text{cost}(S') - \text{cost}(S) = \sum_{i=a}^{t-1} (c_{i+1}(v_i L - v_{i+1}) - c_i v_i) - c_{t-1}v_{t-1} + 2Pv_{t-1}$$

Now $c_i = 2P(L-P)^{t-i-1}$ we have:

$$\begin{aligned}
& \sum_{i=a}^{t-1} \big(2P(L-P)^{t-i-2}(v_iL - v_{i+1}) - 2P(L-P)^{t-i-1}v_i\big) - \\
& \qquad\qquad\qquad\qquad 2P(L-P)^0 v_{t-1} + 2Pv_{t-1} \\
= & \sum_{i=a}^{t-1} \big(2P(L-P)^{t-i-2}(v_iL - v_{i+1} - (L-P)v_i)\big) - 2Pv_{t-1} + 2Pv_{t-1} \\
= & \sum_{i=a}^{t-1} \big(2P(L-P)^{t-i-2}(Pv_i - v_{i+1})\big)
\end{aligned}$$

Because $L \geq P \geq v_{i+1} \geq v_i \geq v_a \geq 1$, we have the above equation is always non negative.

Case: $\hat{i}$ is closed

Since $\hat{i}$ is closed and Lemma 4.4.3, then we must have $v_0 \geq 2$ thus every group must contributes to the cost change.

Note that now in G_0, we have $P + 1 - v_0$ clients needs to pay the extra cost of c_0 and we also close $v_0 - 1$ facilities in S_1. We have a cost change of $+c_0(P + 1 - v_0) - c_0v_0 + c_1(L(v_0 - 1) - v_1) - c_1v_1$ for G_0 and G_1. The cost change of the remaining groups are same as the previous case so we omit the details for the remaining groups. We summarize:

$$\begin{aligned}
\mathrm{cost}(S') - \mathrm{cost}(S) = c_0(P + 1 - v_0) - c_1L + \\
\sum_{i=0}^{t-1} (c_{i+1}(v_iL - v_{i+1}) - c_iv_i) - c_{t-1}v_{t-1} + 2Pv_{t-1}. \quad (4.1)
\end{aligned}$$

Now $c_i = 2P(L-P)^{t-i-1}$ we have:

$$2P(L-P)^{t-2}\left(L(P - v_0) + P(v_0 - P - 1)\right) + \sum_{i=0}^{t-1} \big(2P(L-P)^{t-i-2}(Pv_i - v_{i+1})\big)$$

Note that from previous calculation we saw the sum is non negative, and because $2 \leq v_0 \leq P$ we divide analysis into two cases. First suppose $v_0 < P$, we have $L(P - v_0)$ is at least L and $P(v_0 - P - 1)$ is at least $P(1 - P) \geq -P^2$. Since L can be arbitrarily large then we conclude that if $L > P^2$ then $\text{cost}(S) \leq \text{cost}(S')$.

Second suppose $v_0 = P$, rewrite above equation as following:

$$2P(L-P)^{t-2}\left(L(P-v_0) + P(v_0 - P - 1) + Pv_0 - v_1\right) + \sum_{i=1}^{t-1}\left(2P(L-P)^{t-i-2}(Pv_i - v_{i+1})\right) \quad (4.2)$$

Now because $2 \leq v_0 \leq P$ and $v_0 = P$, we have $2 \leq v_0 = P$ and $v_1 \leq P$:

$$= 2P(L-P)^{t-2}\left(P^2 - P - v_1\right) + \sum_{i=1}^{t-1}\left(2P(L-P)^{t-i-2}(Pv_i - v_{i+1})\right)$$

We see that this case the equation is also non negative thus $\text{cost}(S) \leq \text{cost}(S')$.

Summarizing

We showed that there the instance depicted in Figure 4.3 can not be improved by any test swaps swapping up to P of each type of facilities. We conclude that S is indeed a locally optimum solution and Theorem 4.4.1 follows.

□

Chapter 5

Conclusion

5.1 Summary

The result of this thesis introduced a $(5 + \epsilon)$-approximation algorithm for Budgeted Red-Blue Median. This result is an improvement over the previously known upper bound of 8 from Matroid Median which is a generalization of Budgeted Red-Blue Median [26]. We also discovered that our analysis is tight for the local search algorithm. This shows the strength as well as the limitation of local search technique on this problem. In addition we obtained some corollary results, specifically the lower bound results on extensions of Budgeted Red-Blue Median.

Before digging into the main analysis of our $(5 + \epsilon)$-approximation for Budgeted Red-Blue Median, we first presented how local search approximate k-Median. We followed the analysis of Gupta et al [16] fairly closely until the final step we retold the story in the language of randomization to suit our analysis more. We also explained how local search can be modified so that in the end we indeed have a polynomial time algorithm at our hand. The modification is fairly standard and easily applied to other natural local search algorithm such as Algorithm 4.

We then showed a simple setting of Budgeted Red-Blue Median to explain local search at a very zoomed-in level. With the simple setting we were able to clearly see the cost change of each individual test swap that occurs through out the analysis. This gives us an easy understanding of flow of local search analysis before we dive in to the main analysis.

We presented the part which is considered the main contribution of this work in sections. We first described a procedure to construct structures called blocks. Blocks form a partition of the facilities and helps describe the structure of test swaps. Then we presented the analysis of the cost change of test swaps in cases. The case is defined based on how large is the number of each coloured facilities in a given block. In most cases, we utilized a randomized argument to estimated the expected cost change for each test swap.

Finally, we presented our lower bound results. The lower bound describes the worst case example where a solution that can not be improved by the given local search algorithm may still have a high cost. Typically, the lower bound grows as we increase the number of colors of facilities. For the two colored cases, we see that the lower bound matches our approximation guarantee. Which concludes that the approximation guarantee is tight.

5.2 Future Direction

The most natural open problem related to the results this thesis presents is improving the approximation guarantee for Budgeted Red-Blue Median. That is, we have a $(5+\epsilon)$-approximation guarantee and the hardness result known for this problem is $1+\frac{2}{e}-\epsilon$ for any $\epsilon > 0$ under the assumption $P \neq NP$ [18]. So it is still an open topic to close the gap by either improving the approximation factor or increase the hardness lower bound.

However our result shows that the natural local search algorithm can not do better than the $(5+\epsilon)$-approximation so in order to improve the approximation guarantee, so we are likely having to look into other methods. One promising approach is to look into linear programming rounding technique as it is known that such approach performs better than local search on k-Median([22], [5]).

Bibliography

[1] Sara Ahmadian, Zachary Friggstad, and Chaitanya Swamy. Local-search based approximation algorithms for mobile facility location problems. In *Proc. of SODA*, 2013.

[2] Vijay Arya, Naveen Garg, Rohit Khandekar, Adam Meyerson, Kamesh Munagala, and Vinayaka Pandit. Local search heuristics for k-median and facility location problems. *SIAM J. Comput.*, 33(3):544–562, 2004.

[3] Ivan D. Baev, Rajmohan Rajaraman, and Chaitanya Swamy. Approximation algorithms for data placement problems. *SIAM J. Comput.*, 38(4):1411–1429, 2008.

[4] MohammadHossein Bateni and MohammadTaghi Hajiaghayi. Assignment problem in content distribution networks: Unsplittable hard-capacitated facility location. *ACM Trans. Algorithms*, 8(3):20, 2012.

[5] Jaroslaw Byrka, Thomas Pensyl, Bartosz Rybicki, Aravind Srinivasan, and Khoa Trinh. An improved approximation for k-median, and positive correlation in budgeted optimization. In *Proc. of SODA*, pages 737–756, 2015.

[6] Timothy M. Chan and Sariel Har-Peled. Approximation algorithms for maximum independent set of pseudo-disks. *CoRR*, abs/1103.1431, 2011.

[7] Steven Chaplick, Minati De, Alexander Ravsky, and Joachim Spoerhase. Approximation schemes for geometric coverage problems. *CoRR*, abs/1607.06665, 2016.

[8] Moses Charikar, Sudipto Guha, Éva Tardos, and David B. Shmoys. A constant-factor approximation algorithm for the k-median problem. *J. Comput. Syst. Sci.*, 65(1):129–149, 2002.

[9] Moses Charikar, Samir Khuller, David M. Mount, and Giri Narasimhan. Algorithms for facility location problems with outliers. In *Proc. of SODA, January 7-9, 2001, Washington, DC, USA.*, pages 642–651, 2001.

[10] Vincent Cohen-Addad, Philip N. Klein, and Claire Mathieu. The power of local search for clustering. *CoRR*, abs/1603.09535, 2016.

[11] Nikhil R. Devanur, Naveen Garg, Rohit Khandekar, Vinayaka Pandit, Amin Saberi, and Vijay V. Vazirani. Price of anarchy, locality gap, and a network service provider game. In *Internet and Network Economics, First International Workshop, WINE 2005, Hong Kong, China, December 15-17, 2005, Proceedings*, pages 1046–1055, 2005.

[12] Zachary Friggstad, Mohsen Rezapour, and Mohammad R. Salavatipour. Local search yields a PTAS for k-means in doubling metrics. *CoRR*, abs/1603.08976, 2016.

[13] Zachary Friggstad and Yifeng Zhang. Tight analysis of a multiple-swap heurstic for budgeted red-blue median. In *Proc. of ICALP*, pages 75:1–75:13, 2016.

[14] Matt Gibson, Gaurav Kanade, Erik Krohn, and Kasturi R. Varadarajan. An approximation scheme for terrain guarding. In *Proc. of APPROX*, pages 140–148, 2009.

[15] Inge Li Gørtz and Viswanath Nagarajan. Locating depots for capacitated vehicle routing. In *Proc. of APPROX*, pages 230–241, 2011.

[16] Anupam Gupta and Kanat Tangwongsan. Simpler analyses of local search algorithms for facility location. *CoRR*, abs/0809.2554, 2008.

[17] MohammadTaghi Hajiaghayi, Rohit Khandekar, and Guy Kortsarz. Local search algorithms for the red-blue median problem. *Algorithmica*, 63(4):795–814, 2012.

[18] Kamal Jain, Mohammad Mahdian, and Amin Saberi. A new greedy approach for facility location problems. In *Proceedings of the Thiry-fourth Annual ACM Symposium on Theory of Computing*, STOC '02, pages 731–740, New York, NY, USA, 2002. ACM.

[19] Kamal Jain and Vijay V. Vazirani. Approximation algorithms for metric facility location and k-median problems using the primal-dual schema and lagrangian relaxation. *J. ACM*, 48(2):274–296, 2001.

[20] Ravishankar Krishnaswamy, Amit Kumar, Viswanath Nagarajan, Yogish Sabharwal, and Barna Saha. The matroid median problem. In *Proc. of SODA*, pages 1117–1130, 2011.

[21] Alfred A. Kuehn and Michael J. Hamburger. A heuristic program for locating warehouses. *Management Science*, 9(4):643–666, 1963.

[22] Shi Li and Ola Svensson. Approximating k-median via pseudo-approximation. In *Proc. of STOC*, pages 901–910, 2013.

[23] Mohammad Mahdian and Martin Pál. Universal facility location. In *Proc. of ESA*, pages 409–421, 2003.

[24] Nabil H Mustafa and Saurabh Ray. Improved results on geometric hitting set problems. *Discrete & Computational Geometry*, 44(4):883–895, 2010.

[25] Zoya Svitkina and Éva Tardos. Facility location with hierarchical facility costs. *ACM Trans. Algorithms*, 6(2), 2010.

[26] Chaitanya Swamy. Improved approximation algorithms for matroid and knapsack median problems and applications. In *Proc. of APPROX*, pages 403–418, 2014.

[27] Eduardo Uchoa and Renato Fonseca F. Werneck. Fast local search for steiner trees in graphs. In *Proc. of ALENEX*, pages 1–10, 2010.