The Load-Distance Balancing Problem

Edward Bortnikov[*] Samir Khuller[⋆] Jian Li[⋆] [1] Yishay Mansour[⋄] Joseph (Seffi) Naor[†]

[*] *Yahoo! Research*
Matam Park, Haifa 31905 (Israel)

[⋆] *Department of Computer Science, University of Maryland*
College Park, MD 20742 (USA)

[⋄] *School of Computer Science, Tel-Aviv University*
Tel-Aviv (Israel)
[†] *Computer Science Department, Technion*
Haifa 32000 (Israel)

Abstract

Problems dealing with assignment of clients to servers have been widely studied. However, they usually do not model the fact that the delay incurred by a client is a function of both the distance to the assigned server and the load on this server, under a given assignment. We study a problem referred to as the Load-Distance Balancing problem (or LDB), where the objective is assigning a set of clients to a set of given servers. Each client suffers a delay that is the sum of the distance to its server and the congestion delay at this server, a non-decreasing function of the number of clients assigned to the server.

We address two flavors of LDB – the first one seeking to minimize the *maximum* incurred delay, and the second one targeted for minimizing the *average* delay. For the first variation, we present hardness results, a best possible approximation algorithm, and an optimal algorithm for a special case of linear placement of clients and servers. For the second one, we show the the problem is NP-hard in general and present a 2-approximations for concave delay functions and an exact algorithm if the delay function is convex. We also consider the game theoretic version of the second problem and show the price of stability of the game is at most 2 and at least 4/3.

Keywords: *Approximation Algorithms, Facility Assignment*

[1] To whom the correspondence is to be addressed: Email: lijian@cs.umd.edu, Address: 402, Apt 3, Ridge Rd, Greenbelt, MD, 20770, USA. Phone: 1-240-898-6141.

1 Introduction

The ever-increasing demand for large-scale real-time services to geographically dispersed user populations motivates access providers to deploy advanced services close to the network's edge. Consider, for example, wireless mesh networking (WMN) technologies, which are poised to be a next-generation platform for high-speed Internet access in urban and rural areas [1]. A WMN infrastructure consists of numerous wireless routers, which jointly forward the user traffic to (and from) a limited set of landline gateways. Today, these gateways mainly provide bandwidth sharing of their high-speed access links to the WMN users. Tomorrow, they can be envisioned as a platform for rolling out application-level services with stringent quality-of-service (QoS) requirements. For example, we foresee WMN gateways playing the role of VoIP traffic gateways, media content delivery caches, and even online game servers [5].

In a multi-server setting, *service assignment* problems naturally arise. In this context, each client session must be assigned to an application-level server. Assignment problems have been widely studied in operations research and computer science, and classical problems model the cost of assigning clients to servers as a sum of fixed client-server distances and server (facility) costs. In this context, the servers might or might not have capacities. We identify a need for a more realistic model for describing the *end-user* QoS, e.g., service delay. We model the service delay of a client session as a sum of a *network delay*, incurred by the network connecting the user to its server, and a *congestion delay*, caused by queueing and processing at the assigned server. The delay experienced by each end user is the sum of the distance to the assigned server, and the delay incurred at the server. The *load-distance balancing* problem, or `LDB`, seeks to balance between these two factors, in order to minimize the service delay among all clients. It has two flavors: (1) *maximum* delay minimization (`Min-Max LDB`) and (2) *average* delay minimization (`Min-Avg LDB`).

Summary of Results: We demonstrate that the `Min-Max LDB` problem is NP-hard and present an approximation algorithm with a factor of 2. We also show that the problem is non-approximable with a factor better than 2 for general distance and load functions assuming $P \neq NP$. In addition, we are able to show that for metric spaces (where triangle inequality is satisfied by the distance function) we cannot obtain an approximation factor better than $\frac{5}{3}$ unless $P = NP$. For the special case when the users and the servers are located on a line segment with Euclidean network distances, we present a polynomial time dynamic programming algorithm for this problem.

We show that the `Min-Avg LDB` problem is NP-hard, and can not be approximated within a factor of $(1-\epsilon)\ln n$ for any $\epsilon > 0$ unless $NP \subseteq DTIME(n^{O(\log\log n)})$. For concave delay function, we show the problem is also NP-hard and present a 2-approximation for it. Indeed, the 2-approximation we obtained is a Nash equilibrium for the game theoretic version of the problem in which each client is a selfish player and attempts to minimize its service delay. Hence, the price of stability of the game is at most 2. We also show a lower bound of 4/3 for the price of stability. Moreover, we present a polynomial algorithm, which applies for convex delay functions, and a dynamic-programming solution for the linear setting which has an improved time complexity.

Related Work: The `Min-Max LDB` problem has been introduced in [4]. That work concentrated on solving the problem in a *distributed* setting, in which the servers jointly compute the assignment with partial local data. The protocol of [4] can use any sequential algorithm as a building block. In particular, it can use our algorithm described in Section 3.2, the best possible approximation algorithm. Our paper studies the `LDB` problem in a broader context, and presents new problem variations, algorithms, and hardness results.

Related min-max problems dealing with capacities and facility location were studied before [3, 8, 10]. For example, the capacitated K-center problem [3, 8] asks for K locations to be designated as centers, so as to minimize the maximum distance of a node from its assigned center. In the basic K-center problem, there are no capacities and a center can be assigned an arbitrary number of clients. In the capacitated version each center has a (uniform) load capacity of L, and thus each center can have at most L clients assigned to it. In a sense, this guarantees a bound on the delay of any client, since each is within a distance $O(d^*)$ of its assigned center (d^* is the optimal radius) and cannot suffer a long service time at the assigned center due to the load being at most L. In [8], a 5-approximation on the distance measure was presented for the capacitated K-center problem (improving on a previous bound of 10 [3]) and in addition a $(\frac{2}{c}K, cL, 2d^*)$ solution was presented where $c = 1 + \epsilon$ for any $0 < \epsilon < 1$. This is a solution that uses at most $\frac{2}{c}K$ centers, and allows maximum load to be at most cL, yet provides a 2-approximation for the radius.

In fact, `Min-Avg LDB` is a special case of the universal facility location (`UniFL`) problem [7, 9] (see its definition and the reduction in Section 4.1). The current best known approximation for metric [2] `UniFL` is 6.702 [12] and 1.861 if the facility cost function is concave [7]. For the non-metric `UniFL`, it is known that it is hard to approximate it within a factor of $(1-\epsilon)\ln n$ while whether there is an $O(\ln n)$-approximation is still an open problem [7, 9].

2 Problem Definition

Consider a set of servers $S = \{s_1, \ldots, s_k\}$ and a set of clients $U = \{u_1, \ldots, u_n\}$, so that $k \ll n$. The *network delay* function $D : (U \times S) \to \mathbb{R}^+$ captures the network distance between a client and a server. This function is not necessarily subject to the triangle inequality.

Consider an assignment $\lambda : U \to S$ that maps every client to a server. We assume that each client u assigned to server s adds a unit of *load* on s. We denote the load on s as $\mathcal{L}(\lambda, s) \triangleq |\{u : \lambda(u) = s\}|$. We shorten this to $\mathcal{L}(s)$ when the assignment function is clear from the context. A monotonic non-decreasing *congestion delay* function, $\delta_s : \mathbb{N} \to \mathbb{R}^+$, captures the delay incurred by server s as a function of the number of assigned clients. Different servers can have different congestion delay functions. The service delay $\Delta(u, \lambda)$ of session u in assignment λ is the sum of the two delays:

$$\Delta(u, \lambda) \triangleq D(u, \lambda(u)) + \delta_{\lambda(u)}(\mathcal{L}(\lambda, \lambda(u))).$$

The *maximum* (resp., *average*) cost of an assignment λ is the maximum (resp., average) delay it incurs for a client: $\Delta^M(\lambda(U)) \triangleq \max_{u \in U} \Delta(u, \lambda)$, and $\Delta^A(\lambda(U)) \triangleq \frac{1}{n}\sum_{u \in U} \Delta(u, \lambda)$.

The min-max (resp., min-average) load-distance balancing assignment problem (or `Min-Max LDB` and `Min-Avg LDB` in short) is to find an assignment λ^* such that $\Delta^M(\lambda^*(U))$ (resp., $\Delta^A(\lambda^*(U))$) is minimized. An assignment that yields the minimum cost is called *optimal*. We also study the game theoretic version of `Min-Avg LDB` where each client is a selfish player and aims at minimizing its service delay.

We say an assignment is a Nash equilibrium if no single player can improve its delay by selfishly switching to another server. The objective function is also the average delay. We call this game `Min-Avg LDB Game`. The *price of stability*, defined as the ratio of the delay of the best Nash equilibrium and that of an optimal solution, is used to measure the inefficiency of Nash equilibria. We left the the game theoretic version of `Min-Max LDB` as future work.

[2]The distance function satisfies triangle inequality.

3 Min-Max Load-Distance Balancing

3.1 NP-Hardness

We prove that the `Min-Max LDB` problem is NP-hard. We consider the problem of deciding whether delay Δ^* is feasible, i.e., $\Delta^M(\lambda(U)) \leq \Delta^*$. In what follows, we show a reduction from the classical *exact set cover* (`XSC`) problem. An instance of `XSC` is a collection S of subsets over a finite set U. A solution $S' \subseteq S$ is a cover for U, i.e., every element in U belongs to at least one member of S'. The decision problem is whether there is a cover such that each element belongs to precisely one set in the cover.

Theorem 1 *The* `Min-Max LDB` *problem is NP-hard, even to approximate to a factor strictly less than* 2 *(or* $\frac{5}{3}$ *in metric spaces).*

Proof : Consider an instance of `XSC` in which $|U| = n$, $|S| = k$, and each set contains exactly m elements. The problem is therefore whether there is a cover containing $\frac{n}{m}$ sets.

The transformation of this instance to an instance of `LDB−D` is as follows. In addition to the elements in U, we define a set U' of $M(k - \frac{n}{m})$ dummy elements, where $M > m$. We construct a bipartite graph, in which the left side contains the elements in $U \bigcup U'$ (the clients), and the right side contains the sets in S (the servers). The dummy clients are at distance d_1 from each server. The real clients (elements) are at distance $d_2 > d_1$ from each server (set) that covers them, and at distance ∞ from all the other servers. The capacity of each server for distance d_1 is M, and for distance d_2 is m, i.e., $\delta_s^{-1}(\Delta^* - d_1) = M$, and $\delta_s^{-1}(\Delta^* - d_2) = m$. In other words, the delay at a server for load at most m is $\Delta^* - d_2$ and for load at most M is $\Delta^* - d_1$. It is easy to see that under a feasible assignment, no client's delay exceeds Δ^*.

Each server can cover either (at most) M dummy clients, or any combination of $0 < m' \leq m$ original clients and $m - m'$ dummy clients. If both real and dummy clients are assigned to at least one server, the total number of servers that have real clients assigned to them is $k' > \frac{n}{m}$. All these servers have capacity m, and hence, they serve at most $mk' - n$ dummy clients. The remaining servers can host $M(k - k')$ dummy clients. Hence, the total number of assigned dummy clients is bounded by $M(k - k') + mk' - n = M(k - \frac{n}{m}) - M(k' - \frac{n}{m}) + m(k' - \frac{n}{m}) < M(k - \frac{n}{m})$, that is, the assignment is not feasible. Hence, exactly $\frac{n}{m}$ servers must be allocated to real clients, thus solving the `XSC` instance. The NP-hardness proof is complete.

We simply specify some of the parameters in the above reduction to obtain the imapproximability results. In particular, we show that if there is a solution to the exact cover problem, then there is a solution for the `Min-Max LDB` with cost Δ^*. If there is no solution to the exact cover problem, then all solutions for `Min-Max LDB` have a high cost of ($2\Delta^*$, or $\frac{5}{3}\Delta^*$ in metric spaces).

For the non-metric case, consider the choice $d_1 = 0$ and $d_2 = \Delta^*$. If an element is not a member of a set, the distance to that server is very high. If there is no solution for exact cover, then any collection of $\frac{n}{m}$ sets will leave some element uncovered. The corresponding client will have to be assigned to a server that is also serving $M - 1$ dummy clients. The delay experienced by this client is thus $d_2 + (\Delta^* - d_1) = 2\Delta^*$.

The choice of $d_1 = \frac{\Delta^*}{3}$ and $d_2 = \Delta^*$ preserves the triangle inequality. The distance of a client in C' to a server is either Δ^* or $\frac{5}{3}\Delta^*$. If there is no solution to exact set cover, then the best assignment can have delay no lower than $\frac{5}{3}\Delta^*$. □

3.2 A 2-Approximation Algorithm

We now present 2-approximate solution for `Min-Max LDB`. The algorithm works in phases; in each phase it guesses $\Delta^* = \Delta^M(\lambda^*(U))$, and checks the feasibility of a specific assignment in which neither the network nor the congestion delay exceeds Δ^*, and hence its cost is bounded by $2\Delta^*$. A binary search is performed on the value of Δ^*. A single phase is as follows:

1. Each client u marks all servers s that are at distance $D(u, s) \leq \Delta^*$. These are its feasible servers.

2. Each server s announces how many clients it can serve by computing the inverse of $\delta_s(\Delta^*)$.

3. Define a bipartite client-server graph where an edge specifies that a server is feasible for the client. We need to determine if there is a matching in which the degree of each client is exactly one, and the degree of server s is at most $\delta_s^{-1}(\Delta^*)$. A feasible solution can be found via any flow algorithm.

Theorem 2 *The algorithm computes a 2-approximation of an optimal assignment for* `Min-Max LDB`*.*

Proof : Consider an optimal assignment λ^* with cost Δ^*. It holds that $\Delta_1 = \max_u D(u, \lambda^*(u)) \leq \Delta^*$, and $\Delta_2 = \max_s \delta_s(\mathcal{L}(s)) \leq \Delta^*$. A phase of the algorithm that tests an estimate $\Delta = \max(\Delta_1, \Delta_2)$ is guaranteed to find a feasible solution with cost $\Delta' \leq \Delta_1 + \Delta_2 \leq 2\Delta^*$. □

Since there are at most kn distinct D values, the number of binary search phases is logarithmic in n. The number of phases needed for covering all possible capacity values of server s is $O(\log \delta_s(n))$, which is polynomial in the input size.

3.3 Optimal Assignment on a Line with Euclidean Distances

In this section, we consider the case when the users and the servers are located on a line segment $[0, L]$, and the network delays are Euclidean distances. We show that `Min-Max LDB` is polynomially solvable in this model through dynamic programming.

We start with some definitions. For simplicity of presentation, we assume that every user or server i has a distinct location x_i. The distance between user u and server s is therefore $D(u, s) = |x_s - x_u|$. Assignment λ is called *order-preserving* if for every pair of users u_1 and u_2 such that $x_{u_1} < x_{u_2}$ it holds that $x_{\lambda(u_1)} \leq x_{\lambda(u_2)}$. Otherwise, both λ and every pair (u_1, u_2) for which this condition does not hold are called *order-violating*.

Every order-preserving assignment partitions the line into a series of non-overlapping segments such that every user within segment i is assigned to server s_i. Segment i is located to the left of segment j if and only if $i < j$. Note that s_i is not necessarily located inside segment i.

Theorem 3 *The* `Min-Max LDB` *problem on a line has an order-preserving optimal assignment.*

Proof : Consider an order-violating assignment λ. We show how it can be transformed into an order-preserving assignment that incurs smaller or equal cost.

Since λ is order-violating, there exists a pair of users u_1 and u_2 assigned to servers s_2 and s_1 such that $x_{u_1} < x_{u_2}$ but $x_{s_2} > x_{s_1}$. We transform λ to a new assignment λ' from by switching the assignments of u_1 and u_2, i.e., $\lambda'(u_1) = s_1$ and $\lambda'(u_2) = s_2$. Since this switch does not affect the load on s_1 and s_2, no change is incurred to any user's processing delay. Therefore,

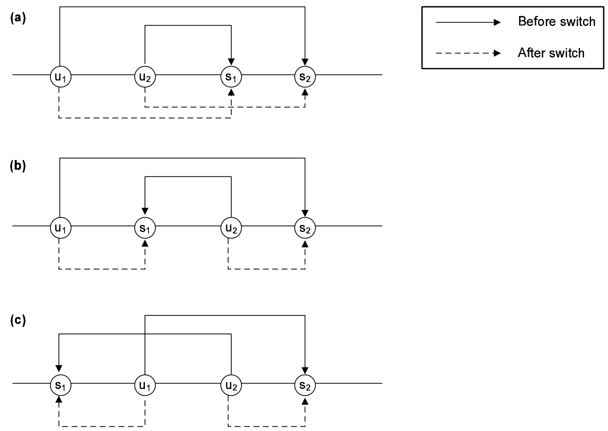

Figure 1: **Switching the assignment of an order-violating pair** (u_1, u_2)**.**

only the network delays incurred to u_1 and u_2 are affected. We therefore need to show that λ' does not incur greater maximum network delay values than λ, that is, we need to show that $\max(D(u_1, s_1), D(u_2, s_2)) \leq \max(D(u_1, s_2), D(u_2, s_1))$. To this end, consider the following cases:

1. $x_{u_1} < x_{u_2} < x_{s_1} < x_{s_2}$ (Figure 1(a)). Then, $D(u_1, s_1) < D(u_1, s_2)$ and $D(u_2, s_2) < D(u_1, s_2)$, hence, $\max(D(u_1, s_1), D(u_2, s_2)) < \max(D(u_1, s_2), D(u_2, s_1))$.

2. $x_{u_1} < x_{s_1} < x_{u_2} < x_{s_2}$ (Figure 1(b)). Then, $D(u_1, s_1) < D(u_1, s_2)$ and $D(u_2, s_2) < D(u_1, s_2)$, hence, $\max(D(u_1, s_1), D(u_2, s_2)) < \max(D(u_1, s_2), D(u_2, s_1))$.

3. $x_{s_1} < x_{u_1} < x_{u_2} < x_{s_2}$ (Figure 1(c)). Then, $D(u_1, s_1) < D(u_2, s_1)$ and $D(u_2, s_2) < D(u_1, s_2)$, hence, $\max(D(u_1, s_1), D(u_2, s_2)) < \max(D(u_1, s_2), D(u_2, s_1))$.

4. $x_{s_1} < x_{u_1} < x_{s_2} < x_{u_2}$. Symmetric to case (2).

5. $x_{s_1} < x_{s_2} < x_{u_1} < x_{u_2}$. Symmetric to case (1).

Thus, we switch the assignment of every order-violating pair of users until an order-preserving assignment is obtained. We conclude that every optimal assignment for `Min-Max LDB` is either order-preserving, or can be transformed into an order-preserving assignment that incurs an equal service delay. □

We now identify the recursive structure of an optimal assignment λ^*. Let $\lambda^*_{i,j}$ for $1 \leq i \leq n$ and $1 \leq j \leq k$ be an optimal assignment for users $\{u_i, \ldots, u_n\}$ that employs servers $\{s_j, \ldots, s_k\}$. We can assign $\ell = 0, \ldots, n-i+1$ leftmost users to server s_j. This assignment defines the maximum delay among the leftmost users. From the optimality of $\lambda^*_{i,j}$, the assignment $\lambda^*_{i+\ell,j+1}$ of the remaining users to the remaining servers is also an optimal one. Hence,

$$\Delta^M(\lambda^*_{i,j}) = \min_{0 \leq \ell \leq n-i+1} [\max(\delta_{s_j}(\ell) + \max_{0 \leq \ell' < \ell} |x_{s_j} - x_{u_{i+\ell'}}|, \Delta^M(\lambda^*_{i+\ell,j+1}))], \tag{1}$$

The boundary conditions are: $\Delta^M(\lambda^* n+1, j) = 0$ (no users), and $\Delta^M(\lambda^* i, k+1) = \infty$ (no servers), for $1 \leq i \leq n$ and $1 \leq j \leq k$. The global optimal assignment cost is $\Delta^M(\lambda^*(U, S)) = \Delta^M(\lambda^*_{1,1})$.

Optimal assignments can be computed through dynamic programming using the above recurrence. An optimal algorithm employs a two-dimensional table `Table`$[1..n+1, 1..k+1]$, where an entry `Table`$[i, j]$ holds the value of $\Delta^M(\lambda^*_{i,j})$, and the number of users assigned to s_j. Note that

$$\max_{0 \leq \ell' < \ell} |x_{s_j} - x_{u_{i+\ell'}}| = \max(|x_{s_j} - x_{u_i}|, |x_{s_j} - x_{u_{i+\ell-1}}|),$$

and hence, the computation of a single entry $\texttt{Table}[i,j]$ incurs $O(1)$ operations for each examined entry $\texttt{Table}[i+\ell, j+1]$. A naive implementation examines $O(n)$ such entries, and therefore, the time complexity of filling the whole table is $O(kn^2)$. This result can be improved by noting that Eq. (1) defines a min-max among the value pairs of $f_{i,j}(\ell) = \delta_{s_j}(\ell) + \max_{0\le\ell'<\ell} |x_{s_j} - x_{u_{i+\ell'}}|$ (a non-decreasing function of ℓ) and $g_{i,j}(\ell) = \Delta^M(\lambda^*_{i+\ell,j+1})$ (a non-increasing function of ℓ). Hence, the min-max is achieved for the value of ℓ for which $f_{i,j}(\ell) - g_{i,j}(\ell)$ is closest to zero. It can be efficiently found through binary search, which yields $O(\log n)$ operations for a single table entry, and $O(kn \log n)$ operations altogether.

4 Min-Average Load-Distance Balancing

In this section, we consider the `Min-Avg LDB` problem. Contrary to the `Min-Max LDB`, the goal is to minimize the *sum* of delays among all users, i.e, $\Delta(\lambda) = \sum_{u\in U} \Delta(u,\lambda)$.

4.1 General Congestion Delay Functions

4.1.1 NP-hardness Proof

We prove the `Min-Avg LDB` problem for general congestion delay functions is NP-hard by reducing from the classical *set cover* problem. In a set cover instance, we have a family $\mathcal{S}$ of subsets of a finite ground set $\mathcal{U}$. A feasible solution to the problem is a collection $\mathcal{S}' \subseteq \mathcal{S}$ of subsets such that for every element $u \in \mathcal{U}$, there exists at least one set $s \in \mathcal{S}'$ with $u \in s$. It is well-known that the set cover problem is NP-hard.

Theorem 4 *The* `Min-Avg LDB` *problem is NP-hard. Moreover, it can not be approximated within a factor of* $(1-\epsilon)\ln n$ *for any constant* $\epsilon > 0$ *unless* $NP \subseteq DTIME(n^{O(\log\log n)})$.

Proof : Suppose we are given a set cover instance with $|\mathcal{U}| = N$ and $|\mathcal{S}| = M$. Let the optimal cover be $\mathcal{S}^*$.

The reduction is as follows. For each set $s \in \mathcal{S}$, we have one server. Let α be any positive integer. All servers have the same delay function $\delta(x) = \begin{cases} 0, & x \le \alpha; \\ 1, & x \ge \alpha+1. \end{cases}$ For each element $u \in \mathcal{U}$, there is a client (which we call *element client*). If $u \in s$, then the distance (the network delay) $D(u,s) = 0$ and $D(u,s) = \infty$ otherwise. For each server s, we also create α "special" clients such that these clients have zero distance to s and infinite distance to other servers (thus in any feasible solution, these clients should be assigned to s). It is not hard to see that every element client u has service delay $\delta(u,\lambda) = 1$ for any reasonable assignment λ. Moreover, for each server s which serves non-zero element clients, all α special clients associated to s have service delay 1. Therefore, an optimal solution for the `Min-Avg LDB` instance uses the minimum number of servers to serve element clients and has cost $\alpha \cdot |\mathcal{S}^*| + N$. This proves the NP-hardness of the problem.

Now, we show the inapproximability result. We need the following result by Feige [6]: For any $\epsilon > 0$, it is impossible to approximate the set cover problem within a factor of $(1-\epsilon)\ln N$ unless $NP \subseteq DTIME(n^{O(\log\log n)})$. Indeed, this result still holds even when we require that $M = N^\delta$ for any $\delta > 0$. Suppose we can get an assignment λ_S having total delay within a factor of $(1-\epsilon)\ln n$ of the optimal one for some $1/2 > \epsilon > 0$. Let $\mathcal{S}' \subseteq \mathcal{S}$ be the collection of sets whose corresponding servers serve non-zero element clients. We can see $\delta(\lambda_S) = \alpha \cdot |\mathcal{S}'| + N \le (1-\epsilon)\log(N + M\alpha)\,(\alpha \cdot |\mathcal{S}^*| + N)$. We assume $M = N^{\epsilon/10}$. By letting $\alpha = N^{1+\epsilon/10}$, we can get

$|\mathcal{S}'| \le (1-\epsilon/2)\log n|\mathcal{S}^*|+O(1)$, which is quite unlikely due to Feige's result. This proves the second part of the theorem. □

4.1.2 The Universal Facility Location Problem

We show `Min-Avg LDB` is a special case of the universal facility location (`UniFL`) problem [7, 9]. In a `UniFL` instance, we are given a set $\mathcal{F}$ of facilities and a set $\mathcal{C}$ of cities. A feasible solution is an assignment λ of the cities to facilities. The facility cost for each facility i depends on the number of cities it serves and is specified by a non-decreasing facility cost function $f_i(.)$. The services cost for each city is equal to the distance between each city and its assigned facility. The goal is to minimize the sum of the facility and service costs, i.e., $\sum_{i\in\mathcal{F}} f_i(L(\lambda,i)) + \sum_{j\in\mathcal{C}} D(j,\lambda(j))$ where $L(\lambda,i)$ is the number of cities that are assigned to i in λ. If the distance function satisfies triangle inequality, we call the problem metric `UniFL` problem, otherwise, we call it non-metric `UniFL` problem.

The reduction is simply as follows. Each facility and city in `UniFL` correspond to a server and a client in `Min-Avg LDB`, respectively. By letting the corresponding facility cost function be $f_s(x) = x \cdot \delta_s(x)$, we can see that a `Min-Avg LDB` instance reduces exactly to a `UniFL` instance with the same optimal cost. Therefore, any approximation for `UniFL` can be carried over to `Min-Avg LDB` with the same ratio. However, all known approximations for `UniFL` are for the metric case. The current best known approximation for metric `UniFL` is 6.702 by Vygen [12] and 1.861 if the facility cost function is concave [7]. We note that $\delta_s(.)$ is concave does not necessarily imply the concavity of the corresponding $f_s(.)$. For the non-metric `UniFL`, it is hard to approximate it within a factor of $(1-\epsilon)\ln n$ for any $\epsilon > 0$ since it also generalizes the set cover problem [6], while whether there is an $O(\ln n)$-approximation is still an open problem [7, 9].

4.2 Concave Delay Funcitons

In many real applications, the delay can roughly be modeled as a concave function of the load. We prove the problem is NP-hard even for a very simple piecewise linear concave delay function. Then we provide a polynomial time 2-approximation. As a byproduct, we show the approximation is a Nash equilibrium for the game theoretic version of the problem, which implies the price of stability of the game is 2.

4.2.1 The NP-hardness

Theorem 5 *The* `Min-Avg LDB` *problem is NP-hard for concave delay function* $\delta(x)$*.*

Proof : The reduction is almost the same as in Theorem 4 except that $\alpha = 1$ and all servers have the same following delay function $\delta(x) = \begin{cases} x, & 0 \le x \le 2; \\ 2, & x \ge 2. \end{cases}$ It is obvious that $\delta(x)$ is a concave function. Using an argument similar to the previous proof, we can see each element client must experience a service delay of 2. Any special client who share common server with any element client has a service delay of 2 and the other special clients experience delay 1. Therefore, an optimal solution has a cost $(M - |\mathcal{S}^*|) + 2|\mathcal{S}^*| + 2N = M + 2N + |\mathcal{S}^*|$ which implies the problem is NP-hard. □

4.2.2 A 2-Approximation Algorithm

We first define the following potential function Φ that maps every assignment into a numeric value. The potential function is similar to the one used in [2].

$$\Phi(\lambda) = \sum_{s \in S} \sum_{x=0}^{\mathcal{L}(\lambda,s)} \delta_s(x) + \sum_{u \in U} D(u, \lambda(u)). \tag{2}$$

The following simple lemma shows the relationship between the potential function value of an assignment λ and the actual delay produced by λ.

Lemma 1 *If $\delta_s(x)$ is non-decreasing and concave for each server s, for any assignment λ, $\frac{1}{2} \cdot \Delta^A(\lambda) \leq \Phi(\lambda) \leq \Delta^A(\lambda)$.*

Proof : By writing $\Delta^A(\lambda) = \sum_{s \in S} \mathcal{L}(\lambda, s)\delta_s(\mathcal{L}(\lambda, s)) + \sum_{u \in U} D(u, \lambda(u))$, we can see the second inequality holds obviously. To see the first inequality, it suffices to show $\frac{1}{2}\ell\delta_s(\ell) \leq \sum_{x=0}^{\ell} \delta_s(x)$ for any $\ell > 0$. Assume that ℓ is odd (The proof for even ℓ is similar and omitted). Due to the concavity of $\delta_s(x)$, we have

$$\sum_{x=0}^{\ell} \delta_s(x) = \sum_{x=0}^{\lfloor \ell/2 \rfloor} (\delta_s(x) + \delta_s(\ell - x)) \geq \sum_{x=0}^{\lfloor \ell/2 \rfloor} (\delta_s(0) + \delta_s(\ell)) \geq \frac{1}{2}\ell\delta_s(\ell).$$

□

We show the problem can be reduced to a min-cost matching computation. We build a bipartite graph where the left part contains n clients, and the right part contains n copies of each server. The weight of edge connecting user u to the i'th copy of server s is $D(u, s) + \delta_s(i)$. Now, we compute a minimum-cost matching such that each clients is matched with one server copy. If client u is matched with some copy of server s, we assign u to s. Moreover, if k copies of s are matched, they should be the first k copies since $\delta_s()$ is a increasing function. Therefore, we can see a min-cost matching corresponds exactly to the assignment with the minimum potential function value.

Let λ' be the assignment that minimizes the potential function Φ, and λ^* be the global optimal solution. From Lemma 1, we have $\Delta(\lambda') \leq 2\Phi(\lambda') \leq 2\Phi(\lambda^*) \leq 2\Delta(\lambda^*)$. Therefore, we have proven the following theorem.

Theorem 6 *There is a polynomial time 2-approximation for* `Min-Avg LDB` *with concave congestion delay functions.*

4.2.3 The Price of Stability of `Min-Avg LDB Game`

In this section, we show the 2-approximation we just presented is actually a Nash equilibrium for `Min-Avg LDB Game`, thus establishing an upper bound of 2 for the price of stability of the game. We also present an example showing an lower bound of the price of stability of 4/3.

The most important property of the potential function is that if a single client u changes its strategy, then the difference between the potential of the new assignment and that of the original one is exactly the change in the delay of u. We formally state it in the following lemma.

Lemma 2 *Consider two assignments, λ and λ', which only differ in the assignment of client u: $\lambda(u) = s$, whereas $\lambda'(u) = s'$. Then, $\Phi(\lambda') - \Phi(\lambda) = \Delta(u, \lambda') - \Delta(u, \lambda)$.*

Proof : It is not hard to see the left-hand side is $\delta_{s'}(\mathcal{L}(\lambda', s')) + D(u, s') - \delta_s(\mathcal{L}(\lambda, s)) - D(u, s)$ which exactly equals to the right-hand side. □

Theorem 7 *The price of stability of the* `Min-Avg LDB` *game with concave $\delta(x)$ is at most 2 and at least 4/3.*

Proof : Let λ' be the assignment that minimizes the potential function Φ and λ^* be the global optimal solution. By Lemma 2, λ' is a Nash equilibrium. By Theorem 6, we can conclude the price of stability is at most 2.

To see the lower bound of 4/3, we consider the following simple instance. Consider two servers s_1 and s_2 with congestion delay functions defined to be $\delta_{s_1}(x) = \begin{cases} x, & 0 \le x \le 2; \\ 2, & x \ge 2. \end{cases}$ and $\delta_{s_2}(x) = 0$ for all $x \ge 0$. We also have two clients u_1 and u_2 with $D(u_1, s_1) = 0, D(u_1, s_2) = +\infty, D(u_2, s_1) = 0$ and $D(u_2, s_2) = 2 + \epsilon$. The global optimum is to assign u_1 to s_1 and u_2 to s_2 and has a cost $3 + \epsilon$. The only Nash equilibrium is to assign both clients to s_1 which has a total cost 4. □

4.3 A Polynomial-time Algorithm for Almost Convex Delay Functions

We now present a polynomial-time algorithm for `Min-Avg LDB` when the function $x\delta_s(x)$ is convex for each server s(most practical congestion delay functions satisfy this requirement). We note that $x\delta_s(x)$ is always convex if $\delta_s(x)$ is convex.

The algorithm reduces the assignment problem to minimum-cost matching in a bipartite graph. The left part contains n clients, and the right part contains n copies of each server (i.e., nk nodes). The cost of connecting user u to the i'th instance of server s is defined as

$$\Delta_i(u, s) = D(u, s) + i\delta_s(i) - (i - 1)\delta_s(i - 1).$$

Intuitively, these costs are *marginal* costs in the assignment, that is, $\Delta_i(u, s)$ is the cost of connecting user u to server s after $i - 1$ other users.

The algorithm computes a minimum-cost matching in the constructed graph (i.e., each user is assigned to exactly one server copy), and turns this matching to a legal assignment by assigning each user to the server it is matched to, regardless of the instance number.

Theorem 8 *The algorithm computes an optimal assignment for* `Min-Avg LDB`*.*

Proof : We first claim that if the copy s_i of server s is utilized by the matching, then all the copies s_j for $j \le i$ are used too. Indeed, suppose by contradiction that user u is matched to some copy s_i ($i > 1$), and s_{i-1} is not used. If u is switched from s_i to s_{i-1}, the matching cost can be reduced by

$$\Delta_i(u, s) - \Delta_{i-1}(u, s) = i\delta_s(i) + (i - 2)\delta_s(i - 2) - 2\delta_s(i - 1),$$

which is a positive value since $x\delta_s(x)$ is a convex function. Hence, the matching's cost can be improved, in contradiction to optimality.

Consider a matching μ in the bipartite graph for which the set of used instances of each server is contiguous, and the corresponding assignment λ for the original problem. We denote the set of users assigned to some instance of server s by $\mu(s)$, and the user assigned to the i'th copy of server s by $\mu_i(s)$. Since the used set is contiguous, the sum of individual matching cost of the users in $\mu(s)$ telescopes to

$$\sum_{i=1}^{|\mu(s)|} \Delta_i(\mu_i(s), s) = |\mu(s)|\delta_s(|\mu(s)|) + \sum_{i=1}^{|\mu(s)|} D(\mu_i(s), s) = \sum_{i=1}^{|\mu(s)|} [D(\mu_i(s), s) + \delta_s(|\mu(s)|)] = \sum_{u:\lambda(u)=s} \Delta(u, \lambda).$$

Hence, the cost of the matching is equal to the cost of an assignment for the original problem. Therefore, since the minimum-cost matching μ^* has the desired property of contiguity, it produces a minimum-cost assignment λ^*. □

4.4 Optimal Assignment on a Line with Euclidean Distances

The fastest known minimum-cost flow algorithm on a graph $G(V, E)$ runs in $O(|E| \log |V|(|E| + |V| \log |V|))$ time [11]. We construct a bipartite graph in which $|V| = O(nk)$ and $|E| = O(kn^2)$, hence the running time is $O(kn^2 \log(nk)(kn^2 + nk \log(nk))) = O(k^2 n^4 \log n)$. In the special case when users and servers are located on a line segment, and network delays are modeled as Euclidean distances, this running time can be significantly improved. Similarly to the `Min-Max LDB` problem, the `Min-Avg LDB` on a line has an order-preserving optimal assignment. Hence, a polynomial time dynamic programming algorithm similar to the one presented in Section 3.3 is applicable in this case. The algorithm's running time is $O(kn^2)$ (in contrast to `Min-Max LDB`, the binary search optimization to reduce the number of operations on a single table entry to $\log n$ cannot be applied).

5 Conclusions and Future Work

We studied two variations of the load-distance balancing (`LDB`) problem, namely, `Min-Max LDB` and `Min-Avg LDB`, which aim to minimize the *maximum* and the *average* delay, respectively. For the first problem, we proved hardness of approximation for general cost functions, and presented the best possible approximation algorithm, as well as an optimal algorithm for the case of linear placement of clients and servers. For the second problem, we showed it is NP-hard and presented approximations for concave delay functions and exact algorithm for convex delay functions. We also show an upper bound of 2 and a lower bound of 4/3 for the price of stability for the game theoretic version of the problem, `Min-Avg LDB Game`.

It would be interesting to achieve an approximation ratio less than 2 for `Min-Max LDB` when the network delay satisfies triangle inequality (note that we proved a lower bound of 5/3). Another open question is whether there is an $O(\ln n)$-approximation for `Min-Avg LDB` with arbitrary non-decreasing congestion delay functions (or even non-metric `UniFL`).

A more general question might be to optimize over the choice of servers, rather than fixing the set of servers. For example, how can we choose a subset of k servers to open, and find an assignment of clients to open servers so as to minimize the average cost, or maximum cost of a client? Alternatively, each server may have a cost, and there may be a budget on the total cost of open servers. These and other questions would be interesting to study.

Acknowledgements

We thank Israel Cidon, Uri Feige, Idit Keidar and Isaac Keslassy for stimulating discussions.

References

[1] I. Akyildiz, X. Wang, and W. Wang. Wireless Mesh Networks: a Survey. *Computer Networks Journal*, 2005.

[2] E. Anshelevich, A. Dasgupta, J. Kleinberg, E. Tardos, T. Wexler, and T. Roughgarden. The price of stability for network design with fair cost allocation. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 295–304, 2004.

[3] J. Bar-Ilan, G. Kortsarz, and D. Peleg. How to allocate network centers. *Journal of Algorithms*, 15:385–415, 1993.

[4] E. Bortnikov, I. Cidon, and I. Keidar. Scalable Load-Distance Balancing. In *International Symposium on Distributed Computing (DISC)*, 2007.

[5] J. Chen, B. Knutsson, B. Wu, H. Lu, M. Delap, and C. Amza. Locality Aware Dynamic Load Management for Massively Multiplayer Games. *ACM Symposium on Principles and Practice of Parallel Programming (PPoPP)*, 2005.

[6] U. Feige. A threshold of $\ln n$ for approximating set cover. *Journal of the ACM*, 45:314–318, 1998.

[7] M. Hajiaghayi, M. Mahdian, and V. Mirrokni. The facility location problem with general cost functions. *Networks*, 42(1):42–47, 2003.

[8] S. Khuller and Y. J. Sussmann. The Capacitated K-Center Problem. *SIAM Journal on Discrete Mathematics*, 13:403–418, 2000.

[9] M. Mahdian and M. Pal. Universal facility location. In *European Symposium on Algorithms (ESA)*, pages 409–421, 2003.

[10] P. B. Mirchandani and R. L. Francis. *Discrete Location Theory.* John Wiley & Sons Inc., 1990.

[11] J. Orlin. A Faster Strongly Polynomial Minimum Cost Flow Algorithm. In *The Annual ACM Symposium on Theory of Computing (STOC)*, 1988.

[12] J. Vygen. From stars to comets: Improved local search for universal facility location. *Operations Research Letters*, 35(4):427–433, 2007.