Online Assignment Algorithms for Dynamic Bipartite Graphs

Ankur Sahai
Department of Computer Science, Indian Institute of Technology, Kanpur.
Email: ankur.sahai@gmail.com

Abstract—This paper analyzes the problem of assigning weights to edges incrementally in a dynamic complete bipartite graph consisting of producer and consumer nodes. The objective is to minimize the overall cost while satisfying certain constraints. The cost and constraints are functions of attributes of the edges, nodes and online service requests. Novelty of this work is that it models real-time distributed resource allocation using an approach to solve this theoretical problem.

This paper studies variants of this assignment problem where the edges, producers and consumers can disappear and reappear or features of their attributes can change over time. Primal-Dual algorithms are used for solving these problems and their competitive ratios are evaluated.

I. MOTIVATION

As more and more data moves to the cloud every day, it becomes important to analyze automated resource management schemes based on online algorithms that make the best use of the available storage while guaranteeing optimal performance to the users. This paper studies the theoretical aspects of the problem of allocating storage to VMs optimally.

The VMs running in a distributed system can be considered as the producers of I/O demands and data-centers as the consumers and this configuration can be visualized as a bipartite graph. This graph is complete as the I/O demand generated by any VM can be assigned to any of the data-centers and vice versa. The capacity of data-centers is equivalent to the capacity of the consumers. The average time per I/O operation or latency between a VM and a data-center can be compared with an edge distance. Failures of edges, consumers and producers can be likened to network link outage, failure of data-center / Storage Area Network (SAN) and VMs respectively and this contributes to the dynamic nature of the bipartite graph.

Allocation of storage demands generated by VMs to data-centers forms the assignment problem in the dynamic bipartite graph. As the I/O demands arrive incrementally and they need to be satisfied instantly, this forms the online part of the problem. Change in edge distances with time, can be visualized as being caused by a mobile user. The assignment without reallocation constraint in problem [13] that prevents reallocation of weights; simulates the practical limitation involved in moving large amounts of data across data-centers within a short period of time. It is to be noted that the available capacity of data-centers is a non-increasing function of time.

This paper presents three points of novelty:

1) Provides a unique theoretical perspective to resource allocation in distributed systems.
2) Analyzes the scenarios where properties of users, resources or the network link between them can change over time.
3) Extends this theoretical approach to mobile users.

II. PROBLEM DEFINITION

This paper aims to analyze online algorithms [4] for dynamically evolving undirected graphs [14, 15, 16]. Given a complete bipartite graph $G = (V, E)$ where, $V$ is a finite set of nodes which consists of producers $i \in P$ and consumers $j \in C$ such that, $V = P \cup C$ and edges $e_{ij} \in E$ with distances $d_{ij}$, where $E = \{e_{ij} \mid i \in P, j \in C\}$. A sequence of online service requests $R = R(t), R(t+1), R(t+2), \ldots$ that are received as input specify either (a) consumer demands (b) failure / restoration of edges, producers and consumers (c) changes in their attributes. This characterizes the dynamic nature of the bipartite graph. Consumer demands act by either changing the edge weights $w_{ij}(t)$ or removing an edge $e_{ij}(t) = 0$. For simplicity, this paper assumes that each producer generates atmost one demand $R_t$ throughout its lifetime and that a unique service request is generated at each time instances $t$. $T$ is the set of all instances at which the online service requests are received $T = \bigcup_t T_t$.

Find an $\alpha$-competitive online algorithm [4] for solving the service requests $R$ that minimizes the distance-weighted sum of edge weights:

$$\sum_{i \in P, j \in C} w_{ij}(t) \cdot d_{ij}(t) \cdot e_{ij}(t) \leq \alpha \cdot \text{OPT}(t), \forall t \in T$$  \hspace{1cm} (1)

where, $\alpha$ is a constant and $\text{OPT}(t)$ is the output of the optimal offline algorithm for the input received in the time interval $[0, t], t \in T$. Such that,

$$\sum_{j \in C} w_{ij}(t) = R_i, \forall i \in P$$  \hspace{1cm} (2)
Equation (2) guarantees that demands generated by producers are satisfied. This will be referred to as producer demand constraint.

$$\sum_{i \in P} w_{ij}(t) \leq M_j(t), \forall j \in C$$  \hspace{1cm} (3)

Equation (3) ensures that consumer capacities at time \( t \in T \) are not exceeded. This will be referred to as consumer capacity constraint.

Dynamic nature of the edges is characterized by,

$$e_{ij}(t) = \begin{cases} 1 & \text{if edge exists, } i \in P, j \in C \text{ at time } t \in T \\ 0 & \text{otherwise} \end{cases}$$  \hspace{1cm} (4)

This paper first analyzes the optimal offline algorithm and then focuses on online algorithms for solving this problem.

### A. Assignment without reallocation

Consider a generalization of the problem in [11] where the weights assigned to edges cannot be reallocated (this is called assignment without reallocation constraint). In this case the weights allocated to edges are a non-decreasing function of time \( w_{ij}(t+1) \geq w_{ij}(t) \) except for edge failures when \( w_{ij}(t+1) = 0, \forall i \in P, j \in C \).

### B. Assignment with varying edge distances

Consider a generalization of assignment without reallocation in [II-A] where edge distances can change over time \( \exists(t, \overline{t}) \in T, t \neq \overline{t} : d_{ij}(t) \neq d_{ij}(\overline{t}) \).

### C. Assignment with node addition / failure or attribute changes

Consider a generalization of assignment without reallocation in [II-A] with addition / failure of producers / consumers, \( \exists(t, \overline{t}) \in T, t \neq \overline{t} : C(t) \neq C(\overline{t}), P(t) \neq P(\overline{t}) \). This paper assumes that when a consumer \( j \in C \) fails the data stored on it is wiped off, \( \sum_{i \in P} w_{ij} = 0 \). This data is recreated by going through the demands generated earlier and is stored on a different consumer.

This section also considers a generalization of this problem where consumer capacities can change over time as specified by the service requests, \( \exists(t, \overline{t}) \in T, t \neq \overline{t} : M_j(t) \neq M_j(\overline{t}), \forall j \in C \).

### D. Offline Assignment with multiple producer requests

Consider a producer that generated multiple requests \( R_{i1}, R_{i2}, \ldots \).

### III. Related Work

Papers that study other theoretical aspects of assigning resources to users incrementally are: online algorithms [7] for the \( k \)-server problems [27], [28], [29], [30], [31], [32], min-flow [7], online matching [4], [6], dynamic assignment [13], bipartite network flow [8] and assignment problem[11], [3]. These are the most relevant results for online resource allocation problem [11].

This problem [11] is also important because it can be used for distributed resource scheduling schemes such as the one used in VMware’s virtualization framework [38], [39]. Virtual Infrastructure using VirtualCenter, a centralized distributed system and recently in VSphere, a cloud OS. The authors of VMware’s Scalable Storage Performance white paper [39] say:

"Latency depends on many factors, including queue depth or capacity at various levels; I/O request size; disk properties such as rotational, seek, and access delays; SCSI reservations; and caching or prefetching algorithms." (39, page 2, para 1)

that the "latency" (time taken to complete I/O request which corresponds to cost of the objective function [11]) depends on "factors" that correspond to the attributes of the producers, consumers and network link considered by this paper [11].

Distributed resource allocation [12], [21], [22], fairness of resource allocation [18], [19] and dynamic load balancing [25], [26], [33] issues have also been studied earlier. A survey of schemes for large scale cloud-computing platforms is presented in [24]. An analysis of the various distributed resource allocation techniques is presented in [17].

### IV. Offline Algorithms

The optimal offline algorithm has to exhaustively look at the available edges. This paper uses Linear Programming (LP) for solving the offline version due to ready availability of LP code [40] that is used for verifying the output for different problem instances.

#### A. LP

Linear Programming [10], [11] is a method used to solve large-scale optimization problems with a set of constraints and an objective function (which has to be either minimized or maximized) both being linear.

The LP formulation for this problem is as follows:

Objective function:

$$\sum_{i \in P, j \in C} d_{ij}(t) \cdot w_{ij}(t), \quad d_{ij}(t) \geq 0, \quad w_{ij}(t) \geq 0$$  \hspace{1cm} (5)

This LP is used for calculating the optimal solution \( \text{OPT}(t) \) for the input received in the time interval \([0, t], t \in T\). As the
demands \( R_i, i \in P \) are non-negative the weight assignments \( w_{ij} \) are also non-negative. Edges that fail \( e_{ij}(t) = 0 \) have their corresponding \( d_{ij}(t) = \infty \) so that, they are not selected.

Constraints:

\[
\sum_{j \in C} w_{ij}(t) \geq R_i, \forall i \in P \tag{6}
\]

Equation (6) represents the producer demand constraint corresponding to (2).

\[
\sum_{i \in P} w_{ij}(t) \leq M_j, \forall j \in C \tag{7}
\]

Equation (7) represents the consumer capacity constraint corresponding to (3). In case the consumer capacity \( M_j(t) \) changes at time \( t \in T \) the corresponding consumer capacity constraint is updated in the new LP formulation at time \( t \).

In addition to this, new constraints corresponding to the existing weight assignment on edges have to be added at each time instance \( t \in T \).

**Theorem 1. (Correctness of LP IV-A)** LP formulation in IV-A produces a valid assignment of weights \( w_{ij} \) on edges \( e_{ij} \) corresponding to the demands \( R \).

**Proof:** Equation (6) guarantees that the total demand generated by producers \( i \in P \) is satisfied. Equation (7) ensures that the capacities of consumers \( j \in C \) are not exceeded. By definition (1) this is a valid assignment of weight on edges. □

**Theorem 2. (Optimality of LP IV-A)** LP formulation in IV-A produces the optimal assignment of weights \( w_{ij} \) on edges \( e_{ij} \) corresponding to the demands in \( R \).

**Proof:** Theorem 1 ensures that this LP produces a valid solution. Since the objective (5) is a minimization function and fractional weights are allowed, it follows that the solution produced by LP is optimal.

For edge failures, the LP formulation IV-A has to be modified by removing the failed edges, adding constraints for the current weight assignments and adding constraints for new set of demands until the next edge failure. □

**B. Primal-Dual**

Primal-Dual algorithms [23] are used for a certain class of optimization problems that involve minimization or maximization of an objective function where there are a finite number of feasible solutions available at each step. These algorithms are based on constructing a dual which is solved in conjunction with the primal. It is used to derive intuitions about the nature of the solution that are implicit in the primal.

Consider the dual [34], [35] of the LP formulation in section IV-A. Let \( y_i \) be the dual variables corresponding to producers \( i \in P \) and \( z_j \) be the dual variables corresponding to the consumers \( j \in C \) then the corresponding dual is:

**Objective function:**

Maximize:

\[
\sum_{i \in P} y_i \cdot R_i - \sum_{j \in C} z_j \cdot M_j, \; y_i \geq 0, \; z_j \geq 0 \tag{8}
\]

Constraints:

\[
y_i - z_j \leq d_{ij}, \forall i \in P, j \in C \tag{9}
\]

Equation (9) suggests that the potential difference between producers and consumers can be at most equal to \( d_{ij} \). This will be referred to as dual potential limit constraint.

Note that by complementary slackness conditions,

\[
w_{ij} > 0 \iff y_i - z_j = d_{ij} \tag{10}
\]

By complementary slackness, weights are allocated \( w_{ij} > 0 \) on edges \( e_{ij}, i \in P, j \in C \) if the potential difference between producer \( y_i \) and consumer \( z_j \) becomes equal to \( d_{ij} \) and vice-versa by (10).

Let \( T(y_i) \) be the set of tight constraints for \( y_i \) such that,

\[
T(y_i) = \{(i, j) : y_i - z_j = d_{ij}\} \tag{11}
\]

Let \( S(y_i) \) be the set of slack constraints for \( y_i \) such that,

\[
S(y_i) = \{(i, j) : y_i - z_j < d_{ij}\} \tag{12}
\]

Consider an unit benefit function of \( y_i \) which measures the increase in dual objective (8)

\[
B(y_i) = R_i - \sum_{j \in (i, j) \in T(y_i)} M_j \tag{13}
\]

**Algorithm 1 Primal-Dual algorithm for Offline Assignment**

\[
\begin{align*}
& y_i \leftarrow 0, \forall i \in P \\
& z_j \leftarrow 0, \forall j \in C \\
& T(y_i) \leftarrow \emptyset \\
& S(y_i) \leftarrow \emptyset \\
& \textbf{while } \exists y_i : B(y_i) \geq 0 \textbf{ do} \\
& \quad y_i : \text{Max}_{y_i} B(y_i) \\
& \quad \delta_1 = \{ \text{Min}_{y_i \in S(y_i)} (y_i, z_j) \mid (i, j) \in S(y_i) \} \\
& \quad y_i \leftarrow y_i + \delta_1 \\
& \quad z_j \leftarrow z_j + \delta_1, \forall j : (i, j) \in T(y_i) \\
& \quad T(y_i) = \{(i, j) \mid y_i - z_j = d_{ij}, \forall j \in C\} \\
& \quad \delta_2 = \text{Min}_{j \in C} \{M_j - \sum_{i \in P} w_{ij} \in (i, j) \in T(y_i)\} \\
& \quad w_{ij} \leftarrow w_{ij} + \delta_2 \\
& \textbf{end while}
\end{align*}
\]

Initializing \( y_i \leftarrow 0, \forall i \in P \) and \( z_j \leftarrow 0, \forall j \in C \) produces a dual feasible solution as the dual potential limit constraint (9) is satisfied. The primal-dual algorithm chooses the \( y_i \) with the highest benefit function \( B(y_i) \) at each step to maximize the increase in value of dual objective function. It then chooses the constraint that is closest to becoming tight and increases the value of \( y_i \) by the amount that is needed to make this constraint tight, \( \delta_1 = \text{Min}_{(i, j) \in S(y_i)} (d_{ij} - (y_i - z_j)) \).
For the set of constraints T that are already tight the corresponding \( z_j \) are also increased by \( \delta_1 \) to maintain tightness. For the constraints that just became tight, the corresponding \( w_{ij} \) are increased to the value of the least available capacity amongst all consumers,

\[
\text{Min}_{j \in C} (M_j - \sum_{i \in P} w_{ij}, (i,j) \in E(y_i)) \tag{14}
\]

**Theorem 3. (Optimality of Algorithm 1)** The Primal-Dual Algorithm [7] reaches the optimal solution for the assignment with reallocation problem in section [7] when it is not possible to increase the potentials in the corresponding dual any further.

**Proof:** This algorithm always produces a dual feasible solution as the dual constraints in potential limit constraint [3] are always satisfied by definition of Algorithm 1. The primal consumer capacity constraints in [7] are always satisfied from the way we increase \( w_{ij} \) from (14). When it is not possible to increase the value of dual objective function the benefit function (13) has a negative value,

\[
B(y_i) < 0 \implies R_i < \sum_{j(i,j) \in E(y_i)} M_j, \forall i \in P \tag{15}
\]

From (14), we know that,

\[
\sum_{j(i,j) \in E(y_i)} M_j = \sum_{j(i,j) \in E(y_i)} w_{ij} \tag{16}
\]

Using (15) and (16) we infer that the demands have been met. This means that primal constraints in (6) have been satisfied and the dual constraints are always satisfied. Thus the solution is optimal.

Complementary slackness (10) is satisfied as we only increase \( w_{ij} \) when the dual constraint is tight. This means that the primal is optimal. □

**Theorem 4. (Complexity of Algorithm 1)** The Primal-Dual Algorithm [7] takes \( O(n^3) \), \( n = |P + C| \) time to complete.

**Proof:** For each producer, it takes \( O(|P|) \) comparisons to calculate the dual variable \( y_i \) with the maximum unit benefit function, \( O(|P| \cdot |C|) \) comparisons (which is equal to the number of dual constraints) to calculate \( \delta_1 \) and \( O(|C|) \) comparisons (which finds the minimum amongst all consumers) to find the \( \delta_2 \). Thus it takes \( O(|P| + |P| \cdot |C| + |C|) \) to execute the while loop in Algorithm 1. For \( |P| \) producers it takes \( O(|P|^2 + |P| \cdot |C| + |P| \cdot |C|) = O(|P|^2 \cdot |C|) = O(n^3) \), \( n = |P + C| \) comparisons. ■

**V. ONLINE ALGORITHMS**

Online algorithms are used for solving problems where the input is received incrementally and partial decisions have to be made at each step. Competitive ratio is used to measure the performance of online algorithm as compared to the optimal offline algorithm that knows the entire input. A study of how randomization can be used to improve the competitiveness of online algorithms is presented in [20].

An \( \alpha \)-competitive online algorithm ALG is defined as follows with respect to an optimal offline algorithm OPT, for a problem \( P \) where, \( I \) is an instance of the problem,

\[
\text{cost}(ALG(I)) \leq \alpha \cdot \text{cost(OPT}(I)) + \beta, \forall I \in P \tag{17}
\]

In equation (17), \( \alpha \) is called the competitive ratio and \( \beta \) can be considered as the startup cost of the algorithm. This paper assumes a startup cost of zero, \( \beta = 0 \).

**A. Assignment without reallocation**

**Algorithm 2: Primal-Dual algorithm for Assignment without reallocation**

\[
y_i \leftarrow 0, \forall i \in P
\]

\[
z_j \leftarrow 0, \forall j \in C
\]

\[
T(y_i) \leftarrow \emptyset
\]

\[
S(y_j) \leftarrow \emptyset
\]

**while** \( \text{producer demand} \) do

\[
\delta_1 = |d_{ij} - (y_i - z_j) | (i,j) = \text{Random}(S(y_i))
\]

\[
y_i \leftarrow y_i + \delta_1
\]

\[
z_j \leftarrow z_j + \delta_1, \forall j : (i,j) \in T(y_i)
\]

\[
T(y_i) = \{(i,j) | y_i - z_j = d_{ij}, \forall j \in C\}
\]

\[
\delta_2 = \min_{j \in C} (M_j - \sum_{i \in P} w_{ij}, (i,j) \in E(y_i)) \tag{18}
\]

\[
w_{ij} \leftarrow w_{ij} + \delta_2 \tag{19}
\]

**end while**

The online adversary [56] produces a sequence of demands that decreases the performance of deterministic algorithms. For a greedy algorithm that selects the minimum cost edge the online adversary produces demands in a non-decreasing order of magnitude denoted \( R_{\text{adv}} \) such that, the highest demands \( R_{\text{MAX}} \) are assigned to edges with the highest distances \( d_{\text{MAX}} \) to maximize the value of objective 5.

\[
R_{\text{adv}} = R_{\text{MIN}} \cdot \ldots \cdot R_{\text{MAX}}.
\]

In response to the service request sequence \( R_{\text{adv}} \) the Algorithm 2 produces a random assignment of weights on edges. After selecting a random edge for weight assignment the corresponding dual variable \( y_i \) is increased to \( d_{ij} \) to satisfy the complementary slackness condition in (10).

Cost (5) of the solution produced by the Algorithm 2 is,

\[
\text{Cost(ALG2)} = \sum R_i \cdot E(d_{e_i}) \tag{18}
\]

\[
E[d_{e_i}] = \sum_{i,j=1}^{\min \{C\}} \frac{d_{e_i}}{j} \leq d_{e_{\text{MAX}}} \cdot \ln |C| \quad \text{(where } H_n \text{ is the nth Harmonic)}
\]

At the first iteration, Algorithm 2 selects an edge from all the available edges, which is \( n = |C| \). By definition of Algorithm 2 after selecting an edge, the weight on the edge
$w_{ij}$ is increased until the capacity of consumer $j$, $M_j$ is reached. So, in the next iteration, one amongst the remaining $n-1$ edges has to be selected. This process is repeated until all consumers are saturated.

Substituting (19) in (18),

$$Cost(ALG2) \leq d_{e_{\text{max}}} \cdot \ln |C| \cdot \sum R_i$$

(20)

The optimal offline solution assigns the lower valued demands to the higher cost edges and saves the lower cost edges for the higher valued demands that arrive later in the sequence. Cost of the solution produced by the optimal offline algorithm is,

$$Cost(OPT) = d_{e_{\text{min}}} \cdot R_{\text{MAX}} + \cdots + d_{e_{\text{max}}} \cdot R_{\text{MIN}}$$

$$\geq d_{e_{\text{min}}} \cdot \sum R_i$$

(21)

From (20) and (21), the competitive ratio (17) in V for Algorithm 3 is,

$$\alpha \leq \frac{d_{e_{\text{max}}}}{d_{e_{\text{min}}}} \cdot \ln |C|$$

(22)

This algorithm also runs in $O(n^3)$ time similar to Algorithm 1 as although choosing a random edge takes constant time, the running time of the algorithm is bounded by the operation that calculates the tight constraints $T(y_i)$.

B. Assignment with varying edge distances

**Algorithm 3** Primal-Dual algorithm for Assignment with varying edge distances

$y_i \leftarrow 0, \forall i \in P$

$z_j \leftarrow 0, \forall j \in C$

$T(y_i) \leftarrow \emptyset$

$S(y_i) \leftarrow \emptyset$

while producer demand

$\delta_1 = |d_{ij} - (y_i - z_j) | \ (i, j) = \text{Random}(S(y_i))$

$y_i \leftarrow y_i + \delta_1$

if edge distance changes then

$z_j \leftarrow z_j + \delta_1 - (d_{ij} - d_{ij}), \forall j : (i, j) \in T(y_i)$

else

$z_j \leftarrow z_j + \delta_1, \forall j : (i, j) \in T(y_i)$

end if

$T(y_i) = \{(i, j) \mid y_i - z_j = d_{ij}, \forall j \in C\}$

$\delta_2 = \min_{j \in C}(M_j - \sum_{i \in P} w_{ij} \cdot \delta_1)$

$w_{ij} \leftarrow w_{ij} + \delta_2$

end while

In Algorithm 3 the primal-dual algorithm accommodates the varying edge distances by adjusting the potentials in potential limit constraint (9) of producers $y_i$ and consumers $z_j$ in order to maintain the tight constraints $T(y_i)$. For the new constraints that become tight $T$, the corresponding primal variables $w_{ij}$ are raised as much as possible.

The online adversary targets the edge with highest weight assignment, $\max_{w_{ij}}$ and increases its distance by a certain amount: $d_{\text{diff}} = d_{ij} - d_{ij}$. Let the existing weight assignment be $A_{\text{ALG3}} = \{e_1 \leftarrow R_1, \cdots, e_n \leftarrow R_n\}$. Say the adversary targets the edge with the highest weight assignment $\max_{w_{ij}}$ and increases the distance of this edge from $d_{ij}$ to $d_{ij}$.

At each iteration of primal-dual Algorithm 3 the expected increase in cost due to varying edge distances $d_{\text{diff}}$ is,

$$E[\text{Cost}_{\text{ALG3}}] = \max_{w_{ij}} \cdot d_{\text{diff}}$$

(23)

At each iteration of OPT IV-A the maximum increase in cost due to varying edge distances is,

$$E[\text{Cost}_{\text{OPT}}] = \max_{w_{ij}} \cdot d_{\text{diff}}$$

(24)

Competitive ratio of Algorithm 7 is the same as the competitive ratio for assignment without reallocation II-A as maximum weight assignments for the Optimal, OPT and Algorithm 3 are equal to the value of maximum demand ($\max_{w_{ij}}(w_{ei}) = \max_{w_{ij}}(w_{ei}) = \max(R_i)$).

This algorithm also runs in $O(n^2)$ time similar to Algorithm 1.

C. Assignment with node addition / failure or attribute changes

**Algorithm 4** Primal-Dual algorithm for Assignment with producer failures

$y_i \leftarrow 0, \forall i \in P$

$z_j \leftarrow 0, \forall j \in C$

$T(y_i) \leftarrow \emptyset$

$S(y_i) \leftarrow \emptyset$

while producer demand

$\delta_1 = |d_{ij} - (y_i - z_j) | \ (i, j) = \text{Random}(S(y_i))$

$y_i \leftarrow y_i + \delta_1$

$z_j \leftarrow z_j + \delta_1 - (d_{ij} - d_{ij}), \forall j : (i, j) \in T(y_i)$

if producer $i$ fails then

$C_i = \{j \mid w_{ij} > 0, \forall j \in C\}$

for $j \in C_i$ do

$w_{ij} = 0, \forall j \in C_i$

$T(y_i) \leftarrow T(y_i) \setminus (i, j)$

end for

$\delta_2 = \min_{j \in C}(M_j - \sum_{i \in P} w_{ij} \cdot \delta_1)$

$w_{ij} \leftarrow w_{ij} + \delta_2$

end while
In problem II-C, the producers/consumers may go down or come up or their demands/capacities can change over time.

1) Producer failure: When a producer \( i \in P \) goes down, the Algorithm 4 first calculates \( C_i = \{ j \mid w_{ij} > 0, \forall j \in C \} \). Then the available capacity \( M_j \) of each consumer on which producer \( i \) demands were assigned is increased by \( w_{ij}, M_j \leftarrow M_j + w_{ij}, \forall j \in C_i \). After this, the corresponding weight assignments are set to zero, \( w_{ij} = 0, \forall j \in C_i \).

The competitive ratio for Algorithm 4 is equal to the competitive ratio for assignment without reallocation \( \text{II-A} \) as the operations corresponding to a producer failure do not affect the competitive ratio.

Additional steps corresponding to the failure of the producer, add \( O(|C|) \) to the time complexity for checking if each consumer contains weight assignments corresponding to producer \( i \in P \). Overall time complexity is \( O(n^2 + |C|) = O(n^2) \) as \(|C| < |V| = n\).

It is to be noted that producers are added online by definition II and the demands generated by them cannot change over time.

Algorithm 5 Primal-Dual algorithm for Assignment with consumer addition

\[
\begin{align*}
y_i &\leftarrow 0, \forall i \in P \\
z_j &\leftarrow 0, \forall j \in C \\
T(y_i) &\leftarrow \emptyset \\
S(y_j) &\leftarrow \emptyset \\
\text{while} \text{ producer demand do} \\
\quad \delta_1 = |d_{ij} - (y_i - z_j) | (i, j) = \text{Random}(S(y_j)) \\
\quad y_i \leftarrow y_i + \delta_1 \\
\quad z_j \leftarrow z_j + \delta_1 - (d_{ij} - d_{ij}), \forall j : (i, j) \in T(y_i) \\
\quad T(y_i) = \{(i, j) \mid y_i - z_j = d_{ij}, \forall j \in C \} \\
\quad \text{if} \text{ consumer j is added then} \\
\quad \quad \text{for} \ i \in P \ \text{do} \\
\quad \quad \quad w_{ij} : \text{Max}_e, \forall w_{ij} > 0 \\
\quad \quad \quad w_{ij} \leftarrow w_{ij} \\
\quad \quad \quad w_{ij} \leftarrow 0 \\
\quad \quad \quad T(y_i) \leftarrow T(y_i) \cup \{(i, j) \} \setminus (i, j) \\
\quad \text{end if} \\
\quad \delta_2 = \text{Min}_{p \in P} (M_j - \sum_{i \in P} w_{ij} : (i, j) \in T(y_i)) \\
\quad w_{ij} \leftarrow w_{ij} + \delta_2 \\
\text{end while} \\
\end{align*}
\]

2) Consumer addition: New consumers will bring with them a new set of edges corresponding to each producer. This can only decrease the cost of existing solution as each producer now has one more edge to choose from for weight assignment.

If distance of this newly added edge \( d_{ij} \) is equal to or more than distances of all the existing edges for which \( w_{ij} > 0 \) then the weight assignment and cost remains the same. Otherwise the weight is transferred from the current edge \( e_{ij} \) with the highest distance to the newly added edge \( e_{ij} \).

The competitive ratio for the primal-dual Algorithm 5 is equal to competitive ratio \( 22 \) of assignment without reallocation problem \( \text{II-A} \) as the cost can only decrease due to newly added edges. For each \( i \in P \) it takes constant time to compare the distance of the newly added edge to the edge on which the weight is currently assigned. This take an additional time of \( O(|P|) \). The overall time complexity is \( O(n^2 + |P|) = O(n^2), |P| < n \).

Algorithm 6 Primal-Dual algorithm for Assignment with consumer capacity decrease

\[
\begin{align*}
y_i &\leftarrow 0, \forall i \in P \\
z_j &\leftarrow 0, \forall j \in C \\
T(y_i) &\leftarrow \emptyset \\
S(y_j) &\leftarrow \emptyset \\
A: \text{while} \text{ Producer demand do} \\
\quad \delta_1 = |d_{ij} - (y_i - z_j) | (i, j) = \text{Random}(S(y_j)) \\
\quad y_i \leftarrow y_i + \delta_1 \\
\quad z_j \leftarrow z_j + \delta_1, \forall (i, j) \in T(y_i) \\
\quad T(y_i) = \{(i, j) \mid y_i - z_j = d_{ij}, \forall j \in C \} \\
\quad \text{if} \ (\overline{M_j} - M_j) < 0 \ \text{then} \\
\quad \quad P_j = \{ i \mid w_{ij} > 0, \forall i \in P \} \\
\quad \quad i = \text{Min}(d_{ij}), i \in P_j \\
\quad \quad w_{ij} \leftarrow w_{ij} - |\overline{M_j} - M_j| \\
\quad \quad w_{ij} \leftarrow w_{ij} + |\overline{M_j} - M_j|, \ T := \text{Min}(d_{ij}), i \neq i \\
\quad \text{end if} \\
\quad \delta_2 = \text{Min}_{p \in P} (M_j - \sum_{i \in P} w_{ij} : (i, j) \in T(y_i)) \\
\quad w_{ij} \leftarrow w_{ij} + \delta_2 \\
\quad \text{generate demand of value } |\overline{M_j} - M_j| \text{ for producer i} \\
\quad \text{goto A} \\
\text{end while} \\
\end{align*}
\]

3) Consumer capacities decrease: The primal-dual Algorithm 6 selects the producer with the minimum distance edge \( (i = \text{Min}(d_{ij}), i \in P_j) \), decreases the weight assigned on edge \( d_{ij} \) by the consumer’s residual weight \( M_{res} \) and assigns \( M_{res} \) on the edge with the next lowest distance. The increase in cost of the dual objective due to this operation is:

\[
E[\text{Cost}_{\text{ALG6}}] = E[\text{Cost}_{\text{OPT}}] = M_{res} \cdot (d_{ij} - \delta_1) \quad (25)
\]

The competitive ratio of Algorithm 6 is the same as the competitive ratio \( 22 \) for assignment without reallocation \( \text{II-A} \) as the optimal algorithm does the same in case of consumer capacity decrease.

Time complexity of Algorithm 6 is the time required to service external demands, time required to find the producers in \( P_j \) and the time to service internal demands \( O(n^2_{\text{external}} + n^2_{\text{internal}} + |P|) = O(n^2), |P| < n \).

4) Consumer failure: Consumer failure can invalidate a current assignment. In this case the weight assignments corresponding to the producers that have weights allocated on the failed consumer \( j \), \( P_j = \{ i \mid w_{ij} > 0, \forall i \in P \} \) are invalidated. This generates internal residual demand corresponding to each
Algorithm 7 Primal-Dual algorithm for Assignment with consumer failures

\begin{align*}
&y_i \leftarrow 0, \forall i \in P \\
&z_j \leftarrow 0, \forall j \in C \\
&T(y_i) \leftarrow \emptyset \\
&S(y_i) \leftarrow 0
\end{align*}

A:

while producer demand do

\begin{align*}
\delta_1 &= \{d_{ij} - (y_i - z_j) \mid (i, j) = \text{Random}(S(y_i))\} \\
y_i &\leftarrow y_i + \delta_1 \\
z_j &\leftarrow z_j + \delta_1 - (d_{ij} - d_{ij}), \forall (i, j) \in T(y_i) \\
T(y_i) &= \{(i, j) \mid y_i - z_j = d_{ij}, \forall j \in C\}
\end{align*}

if consumer \( j \) fails then

\begin{align*}
P_j &= \{i \mid w_{ij} > 0, \forall i \in P\} \\
\text{for } i \in P \text{ do} \\
&\text{generate demand of value } w_{ij} \text{ for producer } i \\
&\text{goto A}
\end{align*}

end if

\begin{align*}
\delta_2 &= \min_{j \in C}(M_j - \sum_{i \in P} w_{ij}(i, j) \in T(y_i)) \\
w_{ij} &\leftarrow w_{ij} + \delta_2
\end{align*}

end while

producer \( i \in P \). This is handled as a regular demand by the Algorithm 7.

The cost of overall solution may increase or decrease depending upon the edge selected for reallocation of weights. If the distance of edge selected is higher than the distance of the edge on which the weight was assigned initially, \( d_{ij} - d_{ij} > 0 \) then the cost increases. If \( d_{ij} - d_{ij} < 0 \) the cost decreases. The cost remains the same if \( d_{ij} = d_{ij} \).

Optimal algorithm will choose the edge with minimum cost available for assigning residual weights. This edge can be found by going through all available edges \(|C|\) for each producer. Thus the Competitive ratio of Algorithm 7 is the same as the competitive ratio \( \frac{22}{22} \) for assignment without reallocation II-A.

Time complexity of Algorithm 7 is equal to that of \( \frac{1}{1} \) as this is a special case of assignment with consumer capacity decrease where consumer capacity is set to zero.

D. Offline Assignment with multiple producer requests

The primal constraint \( \frac{1}{1} \) is extended as follows

\[ \sum_{j \in C} w_{ij}(t) \geq \sum_{i} R_i(t), \forall i \in P, t \in T \quad (26) \]

The unit benefit function (UBF) \( \frac{13}{13} \) is now calculated for each producer demand as follows:

\[ B(y_{it}) = R_i - \sum_{j \in C:t \in T(y_i)} M_j \quad (27) \]

Instead of increasing \( y_i \) (dual variable corresponding to the producer \( i \in P \)) by the entire amount needed to make it tight \( (\delta_1 = d_{ij} - (y_i - z_j)) \), we only increase by the amount proportional to its share in the total producer demand \( (\delta_1 = (R_i - \sum_j R_{ij})/\sum_{i \in P} R_i) \). This ensures that the edge \( e_{ij} \) does not become tight until the producer \( j \in C \) is saturated.

Algorithm 8 Primal-Dual algorithm for Offline Assignment with multiple producer demands

\begin{align*}
&y_i \leftarrow 0, \forall i \in P \\
&z_j \leftarrow 0, \forall j \in C \\
&T(y_i) \leftarrow \emptyset \\
&S(y_i) \leftarrow 0
\end{align*}

while \( \exists y_i : B(y_{it}) \geq 0 \) do

\begin{align*}
y_i &\leftarrow \max_{B(y_{it})} \\
\delta_1 &= \{\min_{(i, j) \in T(y_i)} | (i, j) \in S(y_i)\} \\
y_i &\leftarrow y_i + \delta_1 - (R_i - \sum_{j \in C:t \in T(y_i)} w_{ij}) \\
z_j &\leftarrow z_j + \delta_1 - (i, j) \in T(y_i) \\
T(y_i) &= \{(i, j) \mid y_i - z_j = d_{ij}, \forall j \in C\} \\
\delta_2 &= \min_{j \in C}(M_j - \sum_{i \in P} w_{ij}(i, j) \in T(y_i)) \\
w_{ij} &\leftarrow w_{ij} + \delta_2
\end{align*}

end while

Note that we assume the consumers gets saturated exactly although there could be a producer demand that can only be partly allocated on a given consumer and the remaining part has to be allocated on a different consumer.

VI. CONCLUSION

Variants of the online assignment problem defined in section II can be solved using efficient primal-dual algorithms. Implementing this theoretical approach will improve the performance of automated storage management schemes used in distributed systems.

VII. ACKNOWLEDGEMENT

The author would like to thank Prof. Manindra Agrawal and Prof. Sumit Ganguly at the Department of Computer Science and Engineering at the Indian Institute of Technology (IIT), Kanpur for their valuable suggestions and guidance.

REFERENCES

[1] Kuhn HW, The Hungarian method for the assignment problem, Naval Research Logistics, Quarterly 2, 1955, 8397.
[2] Toroslu IH, Uc Oluk G, Incremental assignment problem, Information Sciences, 177, 6 (March 2007), 15231529.
[3] Mills-Tettey, G. A., Stentz, A. T., and Dias, M. B., The dynamic Hungarian algorithm for the assignment problem with changing costs, Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithms, 954 - 959, 2006.
[4] B. Fuchs, W. Hochstatter, and W. Kern, Online matching on a line, Theoretical Computer Science 332 (1-3), 2005.
[5] Adam Meyerson, Akash Nanavati, Laura Poplawski, Randomized online algorithms for minimum metric bipartite matching, Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithms, 954 - 959, 2006.
[6] S. Khuller, S. Mitchell, and V. Vazirani, On-line algorithms for weighted bipartite matching and stable marriages, Theory of Computer Science 127(2), 1994.
