A New Transportation Problem on a Graph with Sending and Bringing-Back Operations

A New Transportation Problem on a Graph with Sending and Bringing-Back Operations

Tetsuo Asano[1]

Kanazawa University, Kanazawa, and
Japan Advanced Institute of Science and Technology, Nomi, Japan

Abstract. This paper considers a transportation problem which is different from the conventional model. Suppose we are given many storages (nodes) to store multiple kinds of commodities together with roads (edges) interconnecting them, which are specified as a weighted graph. Some storages have surplus and others have shortages. Problem is to determine whether there are transportations to eliminate all of shortages. For transportation we can use a vehicle with the loading capacity at each node. Each vehicle visits one of its neighbors with some commodities which are unloaded at the neighbor. Then, we load some other commodities there, and then bring them back to the original node. How to design such send-and-bring-back transportations to eliminate all shortages is the problem. When we define a single round of transportations to be a set of those transportations at all nodes, whether there is a single round of valid transportations that eliminate all of shortages is our concern. After proving NP-completeness of the problem we present a linear time algorithm for a special case where an input graph is a forest.

Keywords: Forest · linear program · multi-commodity transportation problem · NP-completeness · sending and bringing-back.

1 Introduction

In this paper we consider a new type of a transportation problem which is different from the conventional ones but based on a very natural model. A typical transportation problem in operations research is to find the minimum cost of transporting a single commodity from m sources to n destinations along edges in a given network. This type of problems can be solved using linear programming. Many variations have been considered [2].

The transportation problem to be considered in this paper is defined very naturally. We are given a weighted graph $G = (V, E, w)$, where V is a set of places (nodes) to store many different commodities and E is a set of roads (edges) interconnecting nodes. At each node we can store multiple kinds of commodities. Some storages have surplus and others have shortages. Formally, we specify quantities of commodities associated with a node u by $(w_1(u), w_2(u), \ldots, w_h(u))$, where $w_k(u)$ represents a quantity of the k-th commodity. If $w_k(u) > 0$ then

$w_k(u)$ units of the k-th commodity are stored at node u. On the other hand, $w_k(u) < 0$ means that $|w_k(u)|$ units of the k-th commodity are needed at u.

A vehicle is available at each node for transportation. A transportation between two adjacent nodes u and v using a vehicle at u is done as follows. We load some amounts of commodities at node u, which are specified by $(s_1(u), s_2(u), \ldots, s_h(u))$. After visiting a target node v and unloading them at node v, we load commodities at v, which are specified by $(b_1(u), b_2(u), \ldots, b_h(u))$, and bring them back to the node u. All of vehicles start their nodes simultaneously. The first round of transportations finishes when the transportations are completed at all nodes.

One of our goals is to find a single round of transportations that eliminate all shortages. Of course, it is not realistic to transport a negative quantity of commodity and also to load more than there exists at a node. We also assume that each vehicle has some loading capacity, C. All the vehicles leave their nodes at the same time. Each transportation must be independent of their order. In other words, transportations between adjacent nodes are not synchronous and quantities of commodities are determined before the start time of all transportations. Without such constraint, it would be possible to send commodities no matter how far away it is (by sending some load from node u_1 to u_2 and then we send the load from u_2 to u_3 and so on). This assumption is referred to as the **order independence assumption**, which is a rigid principle in this paper. When we consider a single round of transportations, a transportation from non-negative to non-negative weight or from non-positive to non-positive weight is not effective and thus excluded.

The problem of measuring the Earth Mover's distance is related to this problem, which is a problem of finding the most economic way of transporting soil placed in many different places to holes to fill in. Earth movers are used for transporting the soil. It is possible to use whatever number of earth movers. Problem is to minimize the total cost of transportations. This problem looks hard, but a polynomial-time algorithm using network flow algorithm is known. It is similar to the problem in this paper, but it is different in the following senses: (1) we transport not only one kind of commodity but many kinds, (2) in our case transportations are restricted only between adjacent nodes, (3) we can send commodities from a node u to its adjacent node v and also bring commodities at v back to u, and (4) we discard transportation distances and weight of commodities to carry. Refer to the paper [4] for more detail about the earth mover's distance.

Our first question is, given a weighted graph $G = (V, E, w)$, whether there is a single round of transportations that eliminate all of shortages. Our second question is to find the minimum number of rounds that eliminate all of shortages. The second question looks much harder than the first one. In this paper we first consider the problem in one dimension and show that the first problem can be solved in linear time. The problem in two dimensions is much harder. After proving NP-completeness of the first problem above we present a linear-time algorithm for a special case where an input graph is a forest.

2 Problem Definition

Consider a weighted graph $G = (V, E, w)$, where V is a set of places (nodes) to store commodities and E a set of roads (edges) interconnecting those nodes. At each node we store different kinds of commodities. Some nodes have surplus and others have shortages. Formally, We specify quantities of commodities associated with a node u by $(w_1(u), w_2(u), \ldots, w_h(u))$, where $w_k(u)$ represents a quantity of the k-th commodity. If $w_k(u) > 0$ then $w_k(u)$ units of commodity are stored at node u. On the other hand, $w_k(u) < 0$ means that $|w_k(u)|$ units of the k-th commodity are needed at u. A vehicle for transportation is available at each node.

We specify a transportation using a vehicle at node u by a tuple $(u, \tau(u), (s_1(u), s_2(u), \ldots, s_h(u)), (b_1(u), b_2(u), \ldots, b_h(u)))$, where $\tau(u)$ is a target node and the amounts of commodities to be sent from u to $\tau(u)$ and those brought back from $\tau(u)$ to u are specified by $(s_1(u), s_2(u), \ldots, s_h(u))$ and $(b_1(u), b_2(u), \ldots, b_h(u))$, respectively. Since each vehicle has the loading capacity C, the sum of load must always be within C, that is, for each transportation we must have $\sum_{i=1}^{h} s_i(u) \le C$ and $\sum_{i=1}^{h} b_i(u) \le C$.

All of vehicles leave their nodes at the same time. Due to the order independence assumption, when we send some commodities from u to $\tau(u)$, the amounts of commodities sent from u must be within $(\max(0, w_1(u)), \max(0, w_2(u)), \ldots, \max(0, w_h(u)))$ and also those of commodities brought back from $\tau(u)$ must be within $(\max(0, w_1(\tau(u))), \max(0, w_2(\tau(u))), \ldots, \max(0, w_h(\tau(u))))$.

More formally, for each node u we must have

$$s_k(u) + \sum_{v s.t. \tau(v)=u} b_k(v) \le \max(0, w_k(u)) \quad \text{for each } k = 1, 2, \ldots, h, \text{ and}$$
$$\max(\sum_{k=1}^{h} s_k(u), \sum_{k=1}^{h} b_k(u)) \le C.$$

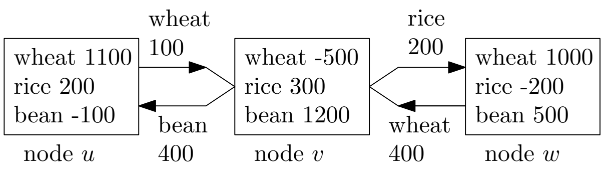

Fig. 1. An example of send-and-bring-back transportations. We send 100 units of wheat at node u, visit v, and bring-back 400 units of bean on the way back from v.

Figure 1 shows an example of transportations. Figure 2 gives another example. In the example, the transportation from node b to node a is characterized by $T_b = (b, a, (0, 30), (60, 0))$ which means we send 0 units of the first commodity and 30 units of the second commodity from the current node b to node a, and then bring-back 60 units of the first one and 0 units of the second one from a to b. Using these transportations we can eliminate all of shortages as shown in (b) in the figure.

Under the definitions above, we consider the following two problems.

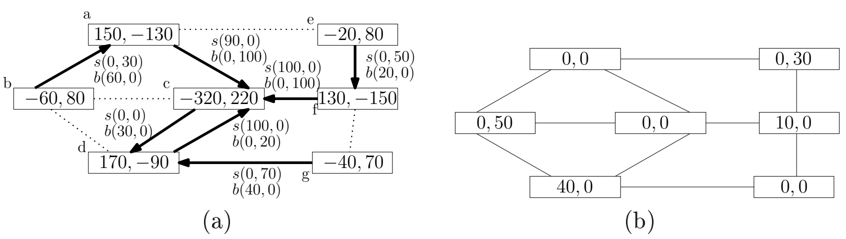

Fig. 2. Transportations necessary to eliminate all the shortages. (a) a single round of transportations (dotted lines are unused edges), and (b) the result after the transportations.

Problem 1: Given a weighted graph, determine whether there is a single round of transportations that eliminate all of shortages. Also, output such a set of transportations if any.

Problem 2: Given a weighted graph, find a single round of feasible transportations that minimize the largest shortage. Also, output such a set of transportations.

2.1 Formal definitions and basic properties

An input instance to our problem is given by a graph $G = (V, E, w)$ in which each node u has h weights $w_1(u), w_2(u), \ldots, w_h(u)$. A transportation using a vehicle at node u is specified by a tuple $(u, \tau(u), (s_1(u), s_2(u), \ldots, s_h(u)), (b_1(u), b_2(u), \ldots, b_h(u)))$ which means that we load $s_i(u)$ units of the i-th commodity at node u for each $i \in [1, h]$ to send them to the target node $\tau(u)$, unload them at $\tau(u)$, and load $b_i(u)$ units of commodity at $\tau(u)$ for each $i \in [1, h]$ to bring them back to u. A single round of transportations is given by a set of transportations at all nodes.

Definition: (Basic Property) A single round of transportations for a graph $G = (V, E, w)$ given by $\mathcal{T} = \{(u, \tau(u), (s_1(u), s_2(u), \ldots, s_h(u)), (b_1(u), b_2(u), \ldots, b_h(u))) | u \in V\}$ is feasible if
(0) $s_i(u) \geq 0$ and $b_i(u) \geq 0$ for each $u \in V$ and $i \in [1, h]$,
(1) the node $\tau(u)$ is adjacent to node u,
(2) $\max(\sum_{i=1}^{h} s_i(u), \sum_{i=1}^{h} b_i(u)) \leq C$ for each $u \in V$, where C is the loading capacity,
(3) there is at least one $i \in [1, h]$ such that $w_i(u) \cdot w_i(\tau(u)) < 0$,
(4) for each $u \in V$, if $w_i(u) > 0$ and $w_i(\tau(u)) < 0$ then $s_i(u) \geq 0$ and $b_i(u) = 0$ for each $i \in [1, h]$,
(5) for each $u \in V$, if $w_i(u) < 0$ and $w_i(\tau(u)) > 0$ then $s_i(u) = 0$ and $b_i(u) \geq 0$ for each $i \in [1, h]$, and
(6) for each $u \in V$ such that $w_i(u) > 0$, $s_i(u) + \sum_{v s.t. \tau(v)=u} b_i(v) \leq w_i(u)$ for each $i \in [1, h]$.

The condition (1) above states that a target node must be selected among those adjacent to node u. The condition (2) states that the sum of load sent from u to $\tau(u)$ and the sum of load brought back from $\tau(u)$ to u must be bounded by the loading capacity. Whenever we transport some units of commodity, we must leave a node having surplus commodity and visit another node of shortage, which is stated in (3). The conditions (4) and (5) state that either $s_i(u)$ or $b_i(u)$ must be 0 for each $i \in [1, h]$. The last condition states that the sum of load of the $i-$th commodity carried out of a node u cannot exceed the original amount $w_i(u)$ of the commodity at u.

Lemma 1. *A feasible single round of transportations $\mathcal{T} = \{(u, \tau(u), (s_1(u), s_2(u), \ldots, s_h(u)), (b_1(u), b_2(u), \ldots, b_h(u)))|u \in V\}$ eliminate all of shortages if $w_i(u) + b_i(u) + \sum_{vs.t.\tau(v)=u} s_i(v) \geq 0$ for each $i \in [1, h]$ and $u \in V$ such that $w_i(u) < 0$.*

Given a single round of transportations $\mathcal{T} = \{(u, \tau(u), (s_1(u),\ s_2(u),\ \ldots,\ s_h(u)), (b_1(u), b_2(u), \ldots, b_h(u)))|u \in V\}$ for a graph $G = (V, E, w)$, we can define a directed graph $G^d = (V, E^d, w)$ where E^d contains a directed edge $(u, \tau(u))$ for each $u \in V$.

Lemma 2. *A single round of transportations $\mathcal{T}$ for a weighted graph $G = (V, E, w)$ is feasible only if the corresponding graph G^d has* **no two cycles** *(including oppositely directed edges between two nodes, called double edges) are contained in one connected component in its underlying undirected graph.*

3 One-dimensional Transportation Problem

3.1 One-commodity problem without capacity constraint

Generally each node stores many different kinds of commodities. First of all, consider only one kind of commodity in each node. Also we assume that a graph is a path, i.e., a linearly ordered array. Suppose n nodes on the horizontal line are numbered like $1, 2, \ldots, n$ from left to right. As input, we assume the quantity of the commodity at node i is given as $w(i)$, where $w(i) < 0$ means a shortage.

Consider a simple case where no loading capacity is assumed. Suppose we are given a sequence $(w(1), w(2), \ldots, w(n))$ of quantities of commodity. There are four cases to consider on the first two nodes. If $w(1) \geq 0$ and $w(2) \geq 0$ then we just discard node 1 since any transportation from node 1 to 2 has no effect. If $w(1) > 0$ and $w(2) < 0$ then we send $w(1)$ units of commodity using the vehicle at node 1 to reduce the shortage at node 2. If $w(1) + w(2) > 0$ then we have to reduce the amount of transportation to prevent the resulting surplus from sending to the right. If $w(1) < 0$ and $w(2) > 0$ then we use the vehicle at node 1 to bring back $\min(|w(1)|, w(2))$ units of commodity from node 2 to 1. If it results in non-negative quantity at node 1, then we proceed to the next step. Otherwise, the sequence is not feasible, i.e., there is no transportation schedule to eliminate the shortage at node 1. If $w(1) < 0$ and $w(2) < 0$ then there is no way to eliminate the shortage at node 1, and hence we just report that the sequence is not feasible and stop.

Lemma 3. *For any instance of a one-dimensional one-commodity transportation problem without constraint on loading capacity we can decide in linear time whether there is a single round of transportations that eliminate all of shortages.*

3.2 One-commodity transportation problem with loading capacity

Next consider some loading capacity C on the total weight to carry by a vehicle. Input is specified by a sequence $(w(1), w(2), \ldots, w(n))$ of n values representing the quantities of commodities. If it contains 0 somewhere, say $w(i) = 0$, then to decide the feasibility we can separate the sequence into two subsequences $(1, \ldots, i-1)$ and $(i+1, \ldots, n)$ since there is no effective transportation from/to the node i of weight 0. If two consecutive values $w(i)$ and $w(i+1)$ are of the same sign, then we can separate the sequence into $(1, \ldots, i)$ and $(i+1, \ldots, n)$ since effective transportation occurs only between two nodes of different signs.

We assume that a sequence starts from a non-negative weight and ends also at a non-negative weight. If not, we add weight 0 at head and/or tail. We also assume that no two consecutive weights have the same sign.

Starting from the initial subsequence $(w(1), w(2), w(3))$, we find optimal solutions that can send the largest amount to the right in two settings. If it is not feasible then there is no feasible solution to the whole sequence. One solution is a feasible solution without constraint and the other with constraint in its last part. Recall that any solution can be represented by a graph on nodes $\{1, 2, 3\}$ with directed edges. We are interested in whether the last part, more exactly, the connected component of the last node 3 in this case in the underlying undirected graph contains double edges. If the last part contains double edges, then this causes some constraint to solutions of further right.

It is easy to compute the best solutions with/without using double edges in the last part. Let them be $\omega_u(3)$ and $\omega_c(3)$, where "u" and "c" represent "unconstrained" and "constrained", respectively. We extend the subsequence by two nodes. Suppose we have computed $\omega_u(k)$ and $\omega_c(k)$. To compute the best solution $\omega_u(k+2)$ there are two ways. One is to use $\omega_u(k)$ and to specify edges among $k, k+1$, and $k+2$ so that double edges are not included there. The other is to use $\omega_u(k)$ and to specify edges within $[k, k+2]$ so that double edges are not included there and also a cut is included between k and $k+1$ or between $k+1$ and $k+2$. The better value among them gives the value of $\omega_u(k+2)$.

Compute of $\omega_c(k+2)$ is symmetric. We compute one solution by using $\omega_u(k)$ and including double edges somewhere among $k, k+1$, and $k+2$. We compute the other one by using $\omega_c(k)$ and including double edges somewhere among $k, k+1$, and $k+2$. The better value among them gives the value of $\omega_c(k+2)$.

Finally, if none of $\omega_u(k+2)$ and $\omega_c(k+2)$ is defined then we have a conclusion that there is no feasible solution. Otherwise, we obtain an optimal feasible solution by taking better one among $\omega_u(n)$ and $\omega_c(n)$.

A formal description of the algorithm is as follows.

```
Algorithm for determining the feasibility of a given sequence.
instance: $(w(1), w(2), \dots, w(n))$.
    If the first and/or last elements are negative we insert 0 as the first and/or
    last elements, respectively.
output: True, if there is a single round of transportations to eliminate all of
    shortages, and False otherwise.
algorithm:
    // Idea is to keep two solutions for each subsequence $(1, 2, \dots, k)$, one
    without constraint and the other with constraint in its last part.
    For the subsequence $(1, 2, 3)$ compute the best solutions, $\omega_u(3)$ in one case
    of no constraint and $\omega_c(3)$ in the other case of constraint, in their last part.
    for $k = 3$ to $n$ step 2 do{
        if $\omega_u(k)$ is defined then{
            $\omega_{uu}(k+2)$ = extend($\omega_u(k)$, unconstrained).
            $\omega_{uc}(k+2)$ = extend($\omega_u(k)$, constrained).
        } else if $\omega_c(k)$ is defined then{
            $\omega_{cu}(k+2)$ = extend($\omega_c(k)$, unconstrained).
            $\omega_{cc}(k+2)$ = extend($\omega_c(k)$, constrained).
        }
        Choose the better one among $\omega_{uu}(k+2)$ and $\omega_{cu}(k+2)$ as $\omega_u(k+2)$.
        Choose the better one among $\omega_{uc}(k+2)$ and $\omega_{cc}(k+2)$ as $\omega_c(k+2)$.
    }
    if at least one of $\omega_u(n)$ and $\omega_c(n)$ is defined and non-negative
    then return True. else return False.
function extend($\omega(k)$, cons){
    if cons = unconstrained then{
        extend the solution $\omega(k)$ by two nodes so that the extended solution is
        not constrained in its last part.
        if it is possible then return the extended solution.
        else return False.
    } else { // cons = constrained
        extend the solution $\omega(k)$ by two nodes so that the extended solution is
        constrained in its last part.
        if it is possible then return the extended solution.
        else return False.
    }
}
```

See Figure 3 as an example. In this example, we have only the unconstrained solution $\omega_u(3)$. Extending it by two nodes, we have only constrained solution $\omega_c(5)$. Using it, we have constrained and unconstrained solutions, $\omega_u(7)$ and $\omega_c(7)$. Finally, we have only constrained solution $\omega_c(9)$.

Lemma 4. *For any instance of a transportation problem with finite loading capacity C we can decide in linear time whether there is a single round of transportations that eliminate all of shortages.*

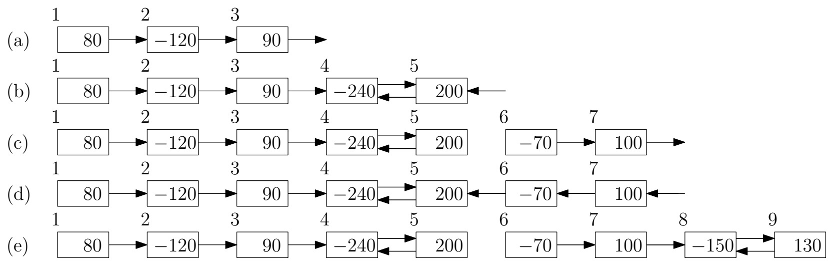

Fig. 3. A behavior of the algorithm. (a) only $\omega_u(3)$ exists, (b) only $\omega_c(5)$ exists, (c) unconstrained solution $\omega_u(7)$, (d) constrained solution $\omega_c(7)$, (e) only constrained solution $\omega_c(9)$ exists, which is a solution.

3.3 Optimization problem

Now, consider the optimization problem to find the minimum shortage we can achieve for a given instance of one-commodity transportation problem. We do not know whether there is a polynomial-time algorithm, but we can design a pseudo-polynomial-time algorithm as follows. Given an instance of one-commodity transportation problem, let $-M$ be the largest shortage. Let $0 < t < M$ be arbitrary number between 0 and M. If we add t to every shortage at each node, we have a modified problem $\mathcal{P}(t)$. More exactly, for each negative weight $w(u) < 0$ we set $w(u) = \min(0, w(u) + t)$ so that no new positive weight is generated. We can decide the feasibility of the problem $\mathcal{P}(t)$ in linear time. If it is feasible then the minimum shortage is at most t. Otherwise, we can conclude that it is beyond t. Using the observation we can find the minimum shortage at any precision by using binary search. The number of iterations is $O(\log M)$.

Lemma 5. *Given a one-dimensional instance of a one-commodity transportation problem, we can determine the minimum shortage in linear time at any precision.*

3.4 Multi-commodity transportation problem

In a multi-commodity transportation problem each node u is characterized by an h-tuple of values $(w_1(i), w_2(i), \ldots, w_h(i))$. If no constraint on loading capacity is assumed, we can implement a transportation for each commodity independently. Therefore, we could apply the algorithm for one-commodity problems to solve a multi-commodity problem.

With constraint on loading capacity, however, we have a trouble. Recall that the algorithm for one-commodity case is based on the fact that a solution for the whole array contains at most two different solutions for the first three nodes. We have found two solutions, one unconstrained and the other constrained. But this is not true anymore for multi-commodity case. For we can design a simple example consisting of four nodes such that any feasible solution for the first three nodes fails to be feasible due to the fourth node.

4 Two-dimensional transportation problem

4.1 NP-completeness

In this subsection we prove NP-completeness of the problem of deciding whether an instance of a general two-dimensional transportation problem is feasible or not, that is, whether there is a single round of transportations that eliminate all of shortages in the instance.

Lemma 6. *The problem of deciding whether, given a weighted graph, there is a single round of transportations with loading capacity C that eliminate all of shortages using sending and bringing-back operations is NP-complete.*

Proof Our proof is based on a reduction from integer partition problem, one of NP-complete problems [1].

Suppose the set $\{a_1, a_2, \dots, a_{2n}\}, a_i > 0, i = 1, \dots, 2n$ is an instance of integer partition, where $\sum_{i=1}^{2n} a_i = 2A$. Then, let U be a collection $\{u_1, u_2, \dots, u_{2n}, u_{2n+1}, u_{2n+2}\}$ with $w(u_i) = A + a_i, i = 1, \dots, 2n$ and $w(u_{2n+1}) = w(u_{2n+2}) = 2n^2A$, and let V be the pair $\{v_1, v_2\}$ with $w(v_1) = w(v_2) = -(2n^2 + n + 1)A$. Consider a bipartite graph $G = (U, V, E)$ where E consists of all edges between U and V. Assume that the loading capacity C is n^2A.

Consider a single round of transportations on the graph G, expressed as a graph G^d. First observation is that each of v_1 and v_2 is incident to exactly one of u_{2n+1} and u_{2n+2} in G^d. If v_1 is incident to none of them, then it is impossible to eliminate the shortage $-(2n^2+n+1)A$ at v_1 even if we send storages of all other nodes to v_1 since $w(v_1)+\sum_{i=1}^{2n} a_i = -(2n^2+n+1)A+2nA+2A < 0$. So, we can assume without loss of generality that v_1 is connected to u_{2n+1} and v_2 to u_{2n+2} in G^d. Moreover, the connection between them must be bi-directional, that is, we have to send n^2A units of commodity from u_{2n+1} to v_1 using the vehicle at u_{2n+1} and also to bring the same amount back to u_{2n+1} since otherwise there is no way to eliminate the large shortage at v_1. So, we can assume that we have bidirectional transportations (double edges in G^d) between u_{2n+1} and v_1 and also ones between u_{2n+2} and v_2. To have a feasible set of transportations we have to send commodities from nodes $u_1, u_2, \dots, u_{2n}$ to either v_1 or v_2 using "send" operations using vehicles at those nodes.

Let U_1 and U_2 be the sets of nodes connected to v_1 and v_2, respectively. Suppose $\sum_{u\in U_1} w(u) < \sum_{u\in U_2} w(u)$. Since the total sum is given by $\sum_{i=1}^{2n+2} w(u_i) = 2n^2A + 2nA + 2A = 2(n^2+n+1)A$, $\sum_{u\in U_1} w(u) \le (n^2+n+1)A$. To eliminate the shortage at v_1 we must have $w(v_1) + \sum_{u\in U_1} w(u) = -(n^2+n+1)A + (n^2 + n + 1)A = 0$. This implies that $\sum_{u\in U_1} w(u) = \sum_{u\in U_2} w(u) = (n^2+n+1)A$.

The above argument implies that if there is a single round of transportations that eliminate all of shortages then there is an integer partition for the set $\{a_1, a_2, \dots, a_{2n}\}$. □

4.2 One-commodity transportation problem on a forest

Consider a special case where we have a single kind of commodity and a graph G is a (undirected) forest composed of trees. We assume that end nodes of each edge have weights of different signs since an edge between two nodes of weights of the same sign has no meaning for transportation as far as we are interested in a single round of transportations.

Consider a simple case first where there is no constraint on the loading capacity. We can deal with each tree in a forest independently. Each tree has at least two **leaf nodes** (of degree 1). Starting from a leaf node u, we traverse a tree until we encounter a **branching node** v of degree ≥ 3. If there is no such branching node, it is just a path, for which we already had an algorithm in Section 3.

Such a path from a leaf node u to branching node v is called a **leaf path**. If a leaf node is adjacent to a leaf branching node, the leaf node is considered as a degenerated leaf path. A **leaf branching node** is a node v of degree ≥ 3 which is incident to at least $deg(v) - 1$ leaf paths, where $deg(v)$ is the degree of v. We create a one-dimensional instance by replacing v in P with an imaginary node $v(P)$ whose weight is 0 if $w(u_k) > 0$ and $+\infty$ otherwise. We can solve the one-dimensional problem using the algorithm in Section 3 which gives us an optimal value x in linear time. If it is infeasible then we have a conclusion that we have no feasible solution. Otherwise, we replace the path with a single node having weight determined by the value of x, that is, x if $w(u_k) > 0$ and $-x$ otherwise.

If we have a leaf branching node whose adjacent set consists of at most one internal node (of degree ≥ 2) and leaf nodes, then we just combine those leaf nodes with the branching node into a single node after a maximal transportation from every such leaf node to the branching node. We repeat the process while each leaf path is feasible. If a single node of weight ≥ 0 is left, then we have a feasible realization. Otherwise, the given tree is not feasible.

A formal description of the algorithm is given below.

Algorithm for deciding the feasibility of a weighted forest: simple case
instance: a weighted forest $G = (V, E, w)$.
assumption: there is no constraint on the loading capacity.
output: True if it is feasible, i.e., there is a single round of transportations that eliminate all of shortages, and **False** otherwise.
algorithm:

```
for each tree $T$ of a forest $G$ do{
    while($T$ contains at least two nodes)do{
        // Dealing with leaf paths
        Find all leaf paths $P_1, P_2, \ldots, P_n$.
        for each leaf path $P_i = (u_{i,1}, u_{i,2}, \ldots, u_{i,k_i}, v_i)$ do{
            if $w(u_{i,k_i}) > 0$ then{
                Replace the last node $v_i$ with a new node $v_n$ of weight 0.
                Formulate a one-dimensional problem $(u_{i,1}, u_{i,2}, \ldots, u_{i,k_i}, v_n)$
            together with a variable $x$ for the edge $(u_{i,k_i}, v_n)$.
            **if** it is not feasible **then return False**.
            Shorten the path $P_i$ to $(v_n, v_i)$ using the node $v_n$ of weight $x$
            and delete all other nodes in the path.
        } **else** {// $w(u_{i,k_i}) < 0$
            Replace the last node $v_i$ with a new node $v_n$ of weight $\infty$.
            Formulate a one-dimensional problem $(u_{i,1}, u_{i,2}, \ldots, u_{i,k_i}, v_n)$
            together with a variable $x$ for the edge $(u_{i,k_i}, v_n)$.
            **if** it is not feasible **then return False**.
            Shorten the path $P_i$ to $(v_n, v_i)$ using the node $v_n$ of weight $-x$
            and delete all other nodes in the path.
        }
    }
    // Dealing with leaf branching nodes
    Find all leaf branching nodes $v_1, v_2, \ldots, v_m$.
    **for** each leaf branching node $v_i$ **do**{
        Let $v_{i,1}, v_{i,2}, \ldots, v_{i,n_i}$ be all leaf nodes adjacent to $v_i$.
        **if** $w(v_i) > 0$ **then** {
            Replace $v_i$ with a new node of weight $w(v_i) + \sum_j w(v_{i,j})$.
            **if** $w(v_i) + \sum_j w(v_{i,j}) < 0$ **then return False.**
        } **else** { // $w(v_i) < 0$.
            Replace $v_i$ with a new node of weight $w(v_i) + \sum_j w(v_{i,j})$.
        }
        Delete all related nodes except for the new node.
    }
  }
}
**return True**.
```

Lemma 7. *Assume no constraint on the loading capacity. Given a weighted forest, it is possible in linear time to determine whether there is a single round of transportations that eliminate all of shortages.*

Next, we consider the constraint on the loading capacity. In the case where no loading capacity is assumed, when we shorten a leaf path to a single node, it induces no constraints to the process of the remaining part. However, in this general case, we have to consider such a constraint. So, we modify the algorithm for a leaf path. We take a leaf path and shorten it to a single node, which becomes a child of a leaf branching node v. When we combine the node v with its children into a single node v', we traverse the tree again from the node v' until we encounter a branching node, which makes a leaf path again. The leaf path may have a constraint in its first part if one of its children is produced with constraint (double edges).

Taking the situations above into account, we consider for a leaf path four different cases depending on whether it has a constraint in its first part and also in its last part. Formally, for a leaf path P_i, we compute four different solutions $\omega_{uu}(P_i), \omega_{uc}(P_i), \omega_{cu}(P_i)$, and $\omega_{cc}(P_i)$, where $\omega_{uc}(P_i)$, for example, is the value of an optimal solution in the case where P_i is unconstrained in its first part but constrained in its last part. It is not so difficult to modify the algorithm for one-dimensional array so that it can compute the four values. The modified algorithm also runs in linear time. Although we omit the detail, we have the following lemma.

Lemma 8. *Given a forest weighted arbitrarily, it is possible in linear time to determine whether there is a single round of transportations that eliminate all of shortages.*

5 Concluding Remarks

We have many open questions. (1) Extension to a multi-commodity problem is not known even for one-dimensional problems. Is there any polynomial-time algorithm? (2) We had an efficient algorithm for minimizing the largest shortage in any precision, which runs in $O(n \log M)$ time where n and M are the size of the array and the largest shortage, respectively. It is efficient in practice, but it also depends on $\log M$. Is there any polynomial-time algorithm which does not depend on M? (3) A more general problem is to determine how many rounds of transportations are needed to eliminate all of shortages. It is more difficult since transportation from a node of positive weight to one of positive weight is effective in two rounds. (4) We assumed that only one vehicle is available, but what happens if two or more vehicles are available at some busy nodes? (5) If vehicles at some nodes are not available, how can we compensate them?

Many other open questions exist although we have no space to list them.

Acknowledgment

This work was supported by JSPS KAKENHI Grant Number JP20K11673. The author would like to thank David Kirkpatrick and Ryuhei Uehara for giving a version of the NP-completeness proof of the transportation problem in two dimensions.

References

1. G.E. Andrews and K. Eriksson, "Integer Partitions," Cambridge University Press, 2004.
2. G.M. Appa, "The Transportation problem and its variants," Oper. Res. Q. , 24, pp:79-99, 1973.
3. T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein, "Introduction to Algorithms (2nd edition ed.), " MIT Press and McGraw-Hill, 2001.
4. S. Peleg, M. Werman, and H. Rom, "A unified approach to the change of resolution: Space and gray-level". IEEE Transactions on Pattern Analysis and Machine Intelligence. 11: pp:739-742, 1989.