Efficient Heuristics for Single-Source Transportation Problems

Pisut Pongchairerks[1*]

[1]Production Engineering Program, Faculty of Engineering, Thai-Nichi Institute of Technology, Bangkok, Thailand.

* Corresponding author. Tel.: +662 763 2600; email: pisut@tni.ac.th

Manuscript submitted July 31, 2014; revised September 27, 2014.
doi: 10.7763/IJAPM.2014.V4.308

Abstract: This paper develops two efficient heuristics to solve the single-source transportation problem, i.e. the special class of the transportation problem where each customer must be served by only one supplier. The proposed heuristics are both started by simplifying the problem into the simpler transportation problem where each customer requires exact one product lot. Then, the first heuristic solves the simplified problem using the transportation algorithm while the second heuristic solves it using the assignment algorithm. The heuristics return an optimal solution or a near-optimal solution for every benchmark instance.

Key words: Single-source transportation problem, assignment problem, transportation problem, heuristic.

1. Introduction

The problem studied in this paper is simulated from the allocation of banknotes from the bank branches to the automated teller machines (ATMs) located in a local government administrative district. The amount of banknotes available in each bank branch is usually much greater than the amount of banknotes needed by each ATM in a local district, and thus each bank branch can serve many ATMs. It is also possible that each bank branch can serve all ATMs in the small local district; however, it may be very costly to serve every ATM by using only one bank branch since each ATM is near a bank branch but far away from another. One ATM is not allowed to receive the banknotes from two or more different bank branches in order to avoid the complexity of the management. The bank branches can be generally defined as suppliers, or sources, and the ATMs can be generally defined as customers, or sinks. This problem is the special class of the transportation problem where each customer must be served by only one supplier. The single-source transportation problem, or single-supplier transportation problem, is the name of this special class of the transportation problem given by literature.

One interesting point of the single-source transportation problem is that this problem can also be viewed as a generalized class of the classical assignment problem. The classical assignment problem consists of a number of suppliers and a number of customers. The assignment problem aims to pair a supplier to a customer for minimizing the total pairing costs. The pairing between suppliers and customers can be changed into other applications such as the pairing between teachers and lecture rooms, etc. In an opposite point of view, the single-source transportation problem is the assignment problem where each supplier can be matched to more than one customer; however, the capacity of each supplier must be greater than or equal to the total demands of its served customers. Based on the characteristics of the problem mentioned above, it can be said that this special problem is an integrative problem between the transportation problem and the assignment problem. This problem is also very similar to the generalized assignment problem; but, in the generalized assignment problem, the customer's demand may be changed when this customer decides to buy products from another supplier. Thus, this problem is also a special class of the generalized assignment problem where each customer has the same demand for every supplier. The statement of the generalized assignment problem is given in literature, e.g. [1].

This paper introduces two heuristic algorithms for this special class of transportation problem. The algorithms both start their processes by simplifying the single-source transportation problem into the transportation problem where each customer has only one product lot as its demand, instead of a number of product units; and each supplier has a number of product lots as its capacity, instead of a number of product units. The first algorithm proposed in this paper then solves the simplified problem using the well-known transportation algorithm, either the stepping-stone method or the MODI method. On the other hand, as the point of view that the simplified problem is to pair one product lot to one customer, it is the assignment problem by this view. The second proposed algorithm then solves the simplified problem using the assignment algorithm, i.e. the Hungarian method. These proposed algorithms will guarantee the optimality for the single-source transportation problem in the conditions either if the number of units demanded by every customer is exactly same, or if every supplier alone has the number of units sufficient to satisfy all customers' demands.

2. Literature Review

The single-source transportation problem is firstly introduced by [2]. The literature proposes the 0-1 integer programming model as given in the model (1) – (4). The problem consists of a set of suppliers $I = \{1,\dots, i,\dots, m\}$ with predefined capacity b_i and a set of customers $J = \{1,\dots, j,\dots, n\}$ with predefined demand r_j of identical product' units. The objective is to minimize the total transportation costs Z subject to the following constraints: (1) the total amount of units shipped from each supplier must not exceed its capacity and (2) each customer must receive its product units from exactly one supplier. The distribution cost of all unit product units from the supplier i to the customer j equals to c_{ij}. Defining $x_{ij} = 1$ if the product units are all shipped from the supplier i to the customer j whereas $x_{ij} = 0$ if none of the product units is shipped from the supplier i to the customer j. De Maio and Roveda in [2] present an implicit-enumeration approach to solve the problem.

$$\min \quad Z = \sum_{i\in I}\sum_{j\in J} c_{ij}x_{ij} \tag{1}$$

subject to

$$\sum_{j\in J} r_j x_{ij} \le b_i \qquad \forall i \in I = 1,\dots,m \tag{2}$$

$$\sum_{i\in I} x_{ij} = 1 \qquad \forall j \in J = 1,\dots,n \tag{3}$$

$$x_{ij} \in \{0,1\} \qquad \forall i \in I, \forall j \in J \tag{4}$$

As mentioned above, the problem presented by [2] can be classified as the special class of the generalized assignment problem where the demand of each customer is the same when the customer changes to buy the products from a different supplier. Thus, the programming model of the generalized assignment problem is different from the model (1)–(4) in the Constraint (2). In the generalized assignment problem, the r_j must be replaced by r_{ij} in Constraint (2), where r_{ij} is the number of product's units required by Customer j for Supplier i. The surveys of the generalized assignment problem are given in [1], [3], [4].

Srinivasan and Thompson in [5] later transform the 0-1 integer programming model given by [2] into an integer programming model whose form is very close to the classical transportation problem by assigning $y_{ij} = r_j x_{ij}$ and $d_{ij} = c_{ij} \div r_j$ for every i and j. They then use the model in their branch-and-bound algorithm. Defining $y_{ij} = r_j$ if the product units are all shipped from the supplier i to the customer j whereas $y_{ij} = 0$ if none of the product units is shipped from Supplier i to Customer j. The integer programming model (5)–(8) are slightly different from the original model of [5] in that the model of [5] provides the equation in Constraint (6) by adding a slack to make the model in standard form.

$$\min \quad Z = \sum_{i\in I}\sum_{j\in J} d_{ij} y_{ij} \tag{5}$$

subject to

$$\sum_{j\in J} y_{ij} \le b_i \qquad \forall i \in I = 1,\ldots,m \tag{6}$$

$$\sum_{i\in I} y_{ij} = r_j \qquad \forall j \in J = 1,\ldots,n \tag{7}$$

$$y_{ij} \in \{0, r_j\} \qquad \forall i \in I, \forall j \in J \tag{8}$$

The research from [6] proposes two heuristics and a branch-and-bound algorithm to solve the single source transportation problem. The two heuristics differ only in the order in which customers are selected to be assigned to suppliers. The first heuristic calculates regret similar to those of the Vogels Approximation Method (VAM) starting solution method [7] for the transportation problems while the second heuristic selects the customers in the order of non-increasing demand sizes.

A full research report of [8], published partly in [9] and [10], compares the performances of four branch-and-bound algorithms for the single-source transportation problem. The first branch-and-bound algorithm is based on the integer programming model (5)–(8). The algorithm makes the relaxed problem by relaxing Constraint (8) into $0 \le y_{ij} \le r_j$ and uses the binary search tree structure to divide the relaxed problem into subproblems. The relaxed problem is the transportation problem, so that it can be solved by the MODI method. The second branch-and-bound algorithm is similar to the first branch-and-bound algorithm, but it instead uses the single assignment search tree structure to divide the relaxed problem into subproblems. The third and fourth branch-and-bound algorithms are based on the integer programming model (1)–(4). To generate the relaxed problem, they relax Constraint (2) to be without the constraint of capacity limit. The algorithms solve the relaxed problem by Hungarian method. However, Hungarian method can solve only one-to-one pairing; to make Hungarian method usable for the relaxed problem, each supplier must be divided into a number of sub-suppliers; this number of sub-suppliers equals the maximum number of customers which can be served by this supplier, sorting from the customer with lowest demand. The third branch-and-bound algorithm uses the binary search tree structure to divide the relaxed problem into subproblems, while the fourth branch-and-bound algorithm uses the single assignment search tree structure instead. Based on all problem instances given by [8], the third and the first branch-and-bound algorithms perform best and second-best respectively.

For the generalized assignment problem which is a generalized class of the single-source transportation problem, a branch-and-bound algorithm is developed in [3] to solve the generalized assignment problem by using a series of binary knapsack to determine the bounds. Some approximation algorithms are developed for this problem as follows. Cohen et al. in [11] propose an efficient approximation. They use the technique based on the combinatorial translation of any algorithm for the knapsack problem into an approximation algorithm. Romeijn and Morales in [12] propose greedy heuristics based on the approach for the multi-knapsack problem. Many genetic algorithms have been used for solving the generalized assignment problem, e.g., [13]-[15]. Other meta-heuristic algorithms involved to the generalized assignment problem are tabu search algorithm in [16], and ant colony optimization algorithm in [17].

3. Proposed Algorithms

At the initial step of the two heuristic algorithms proposed here, the single-source transportation problem, or single-supplier transportation problem, must be transformed into a simpler problem by using product lots instead of product units; a supplier may have many product lots but a customer must require only one product lot. All product units needed by a customer will be distributed from a supplier as a whole in one product lot. It means that in the simplified problem each customer requires exact one product lot which contains the number of product units given in the original problem. The distribution cost per lot from a supplier to each customer is the multiplication of the number of product units of the customer and the distribution cost per unit from the supplier to the customer.

As mentioned, the number of product lots for each customer is only one product lot. Now, in the side of the suppliers, each supplier will have a specific number of product lots available to supply. To define the number of product lots for each supplier may require both science and art in making the decision. Assigning too large number of product lots tends some suppliers to supply over their capacities, while assigning too small number of product lots tends some customers to not receive any product lot. This paper proposes the method to find the proper number of product lots for each supplier as shown in Algorithm 1. For each supplier, Algorithm 1 must be repeated once. Algorithm 1 provides the maximum number of product lots for each supplier such that the case that the size of a product lot is smaller than a customer's demand does not occur.

Algorithm 1. Assigning number of product lots for each supplier

Step 1. Sort all given customers into the list, starting from the customer which has the highest demand to the customer which has the smallest demand.

Step 2. Count the maximum number of customers which can be served by the available capacity of the supplier being considered; that is the sum of the demands of these customers is not greater than the supplier's capacity, but the sum of the demands of these customers plus the demand of the next customer in the list is greater than the supplier's capacity. The customer can be counted only if all other customers above it in the list have been counted.

Step 3. Let the number of product lots available for the supplier being considered equals to the result of counting value.

The procedure of the first heuristic algorithm is given in Algorithm 2. This algorithm transforms the single-source transportation problem into the classical transportation problem, as a simpler problem that aims to supply the product lots instead of product units; and then solves it using the well-known transportation model, i.e. MODI method or stepping stone algorithm.

Algorithm 2. The procedure of the first heuristic algorithm.

Step 1. Let every customer demand only a single product lot, and use Algorithm 1 to assign the number of product lots for every supplier, as its capacity.

Step 2. Transform the original problem into the transportation problem as shown in the programming model (9)–(12) where x_{ij} is the number of product lots shipped from Supplier i to Customer j, c_{ij} is the distribution cost per product lot from Supplier i to Customer j, e_i is the total number of product lots available in Supplier i taken from Algorithm 1. The distribution cost per product lot from Supplier i to Customer j, c_{ij}, is the distribution cost per unit from Supplier i to Customer j, d_{ij}, multiplied by the number of all product units needed by Customer j, r_j; that is $c_{ij} = d_{ij} \times r_j$.

Step 3. Check whether the simplified transportation problem is feasible or not by comparing the total number of product lots of all suppliers with the number of all customers. If the total number of product lots of all suppliers is less than the number of all customers, the problem is infeasible; it means some customers will not receive any product lot. Let repair it to feasibility by assigning the highest-demand customer to the supplier that requires the lowest unit cost and has capacity not less than the demand of this customer. If there are two or more supplier-customer pairs satisfying this property, select one of them arbitrary. Ignore the chosen customer from the problem as well as reduce the capacity of the chosen supplier by the demand of the chosen customer; then repeat from Step 1. Otherwise, the problem is feasible, move to Step 4.

Step 4. Use MODI method or the stepping-stone algorithm to solve the simplified problem. Transform the solution of the simplified problem into the solution of the original problem by the interpretation as follows: a particular customer receives a product lot from a particular supplier in the simplified problem's solution means that this customer receives the number of all demanded product units from the mentioned supplier.

$$\min \quad Z = \sum_{i \in I} \sum_{j \in J} c_{ij} x_{ij} \tag{9}$$

subject to

$$\sum_{j \in J} x_{ij} \le e_i \qquad \forall i \in I = 1, \ldots, m \tag{10}$$

$$\sum_{i \in I} x_{ij} = 1 \qquad \forall j \in J = 1, \ldots, n \tag{11}$$

$$x_{ij} \in \{0,1\} \qquad \forall i \in I, \forall j \in J \tag{12}$$

The procedure of the second heuristic algorithm is given in Algorithm 3. This algorithm transforms the single-source transportation problem into the classical assignment problem and solves it using the well-known assignment model, Hungarian method. This is the assignment problem which needs to pair a product lot to a customer.

Algorithm 3. The procedure of the second heuristic algorithm.

Step 1. Let every customer demand only a single product lot, and use Algorithm 1 to assign the number of product lots for every supplier, as its capacity.

Step 2. Numbering all product lots of all suppliers, starting from 1 through l, where $l = e_1 + e_2 + \ldots + e_m$. Notice that the product lot # 1 through the product lot # e_1 belong to Supplier 1, the product lot # $[e_1 + 1]$ through the product lot # $[e_1 + e_2]$ belong to Supplier 2, and so on. Transform the problem into an assignment problem as shown in the programming model (13) – (16) where x^*_{kj} = 1 if the product lot # k is shipped to Customer j; otherwise x^*_{kj} = 0. c^*_{kj} is the cost of delivering the product lot # k to Customer j and its value is exactly same to the distribution cost per product lot of the particular supplier, which is the owner of product lot # *k*, to Customer *j*.

Step 3. Check whether the assignment problem is feasible or not by comparing the number of all product lots, or *l*, with the number of all customers, or *n*. If the total number of product lots of all suppliers is less than the number of all customers, or $l < n$, the problem is infeasible; it means some customers can not receive any product lot. Let repair it to feasibility by assigning the highest-demand customer to the supplier that requires the lowest unit cost and has capacity not less than the demand of this customer. If there are two or more supplier-customer pairs satisfying this property, select one of them arbitrary. Ignore the chosen customer from the problem as well as reduce the capacity of the chosen supplier by the demand of the chosen customer; then repeat from Step 1. Otherwise, the problem is feasible, move to Step 4.

Step 4. Use the Hungarian algorithm to solve the simplified problem. Transform the solution of the simplified problem into the solution of the original problem by the interpretation as follows: a particular customer receives a product lot from a particular supplier in the simplified problem's solution means that this customer receives the number of all demanded product units from the mentioned supplier.

$$\min \quad Z = \sum_{k\in K}\sum_{j\in J} c^*_{kj} x^*_{kj} \tag{13}$$

subject to

$$\sum_{j\in J} x^*_{kj} = 1 \qquad \forall k \in K = 1,\dots,l \tag{14}$$

$$\sum_{k\in K} x^*_{kj} = 1 \qquad \forall j \in J = 1,\dots,n \tag{15}$$

$$x^*_{kj} \in \{0,1\} \qquad \forall k \in K, \forall j \in J \tag{16}$$

To clarify the processes of the two proposed algorithms, an example is given as follows. There are two suppliers and five customers. Supplier 1 has the capacity of 20 units and Supplier 2 has the capacity of 10 units. The number of product units needed by each customer is provided in Table 1 and the distribution cost per unit from a supplier to each customer is shown in Table 2. The problem in this example needs to find the distribution plan to distribute the product units to the customers based on the condition that each supplier must receive its product units from only one supplier.

The demonstration of using the first algorithm to solve the single-source transportation problem in the above example is given via Algorithm 2 as follows. In Step 1, the order of the customers sorted by their demands from highest to lowest is Customers 1, 4, 2, 3 and 5 as an ordered list. Supplier 1 which has a capacity of 20 can serve Customers 1, 4 and 2 which altogether has their demands of 18. Thus, as the supplier can serve the three first customers in the ordered list, the number of product lots of Supplier 1 is thus 3 lots in order to avoid the case that the lot size is smaller than a customer demand, i.e. $e_1 = 3$. Supplier 2 which has a capacity of 10 can serve Customer 1 which has a demand of 9 only. Notice that the Supplier 2's capacity, i.e. 10, is not less than the demand of Customers 1, i.e. 9, but is less than the sum of the demands of Customers 1 and 4 which is 9 + 5 = 14. Thus, as it can serve only the first customer in the ordered list, the number of product lots of Supplier 2 is 1 lot, i.e. $e_2 = 1$, in order to avoid the case that the lot size is smaller than a customer demand. The distribution cost per product lot from a supplier to a customer is the demand of the customer multiplied by the distribution cost per unit as given in Table 3. The simplified problem is expressed in Fig. 1.

Table 1. Demands of Customers (Units)

	Customer 1	Customer 2	Customer 3	Customer 4	Customer 5
Demand	9	4	3	5	3

Table 2. Distribution Cost Per Product Unit ($ Per Unit)

	Customer 1	Customer 2	Customer 3	Customer 4	Customer 5
Supplier 1	2	3	1	1	4
Supplier 2	3	2	4	3	1

Table 3. Distribution Cost Per Product Lot ($ Per Lot)

	Customer 1	Customer 2	Customer 3	Customer 4	Customer 5
Supplier 1	18	12	3	5	12
Supplier 2	27	8	12	15	3

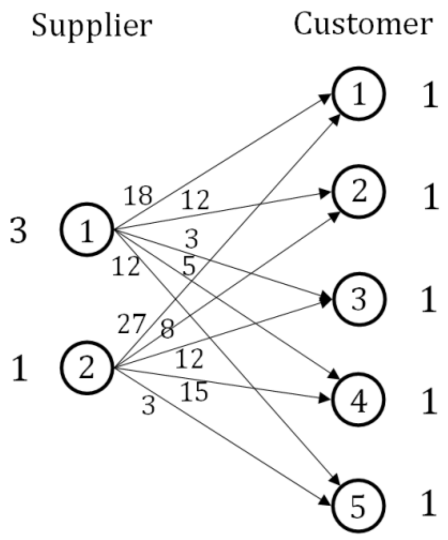

Fig. 1. Simplified transportation model of example case in first round of Algorithm 2.

As shown in Fig. 1, the total number of product lots available in all suppliers, i.e. 3 + 1 = 4, is less than the number of all customers, i.e. 5, the simplified problem is infeasible as one customer cannot receive a product lot. To repair it, Customer 1 is paired to Supplier 1 because Customer 1 has the highest demand in the original problem and Supplier 1 offers the lowest unit cost for this customer. After that, repeat Algorithm 2 from Step 1 again, and keep in mind that the demand of Customer 1 has been fulfilled by Supplier 1 already. In the second round of Step 1, the order of the customers sorted by their demands from highest to lowest is Customers 4, 2, 3 and 5. Supplier 1 which has its remaining capacity of 11 can serve Customers 4 and 2 which altogether have total demands of 5 + 4 = 9 units. Thus, as the supplier can serve the two first customers, the number of product lots of Supplier 1 is 2 lots, i.e. $e_1 = 2$. Supplier 2 which has a capacity of 10 can serve Customers 4 and 2 which altogether have total demands of 9 units. Thus, as it can serve the two first customers, the number of product lots of Supplier 2 is 2 lots, i.e. $e_2 = 2$. Fig. 2 shows the simplified model in the second round of Algorithm 2. Since the total number of product lots available by all suppliers, i.e. 4, is not less than the number of all customers, i.e. 4, the simplified problem is feasible. Then, use the MODI method to solve this simplified problem and the optimal solution for the simplified problem is to deliver a product lot from Supplier 1 to Customer 3, a product lot from Supplier 1 to Customer 4, a product lot from Supplier 2 to Customer 2, and a product lot from Supplier 2 to Customer 5; in addition with a prior decision, a product lot is delivered from Supplier 1 to Customer 1. The solution value is 3 + 5 + 8 + 3

+ 18 = $37. Note that the optimal solution of the simplified problem may or may not be the optimal solution of the original problem.

This solution of the simplified problem is interpreted into the original problem's solution as follows: 1 lot from Supplier 1 to Customer 3 = 3 units from Supplier 1 to Customer 3, 1 lot from Supplier 1 to Customer 4 = 5 units from Supplier 1 to Customer 4, 1 lot from Supplier 2 to Customer 2 = 4 units from Supplier 2 to Customer 2, 1 lot from Supplier 2 to Customer 5 = 3 units from Supplier 2 to Customer 5, and 1 lot from Supplier 1 to Customer 1 = 9 units from Supplier 1 to Customer 1, with the solution value of $37.

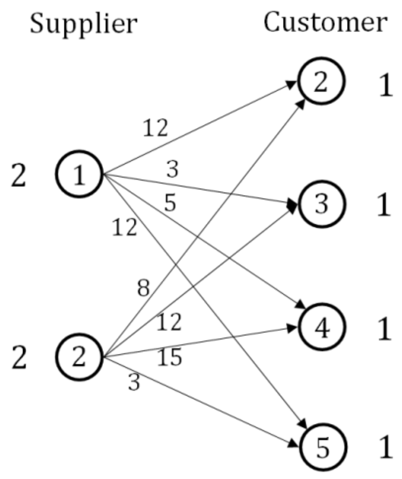

Fig. 2. Simplified transportation model of example case in second round of Algorithm 2.

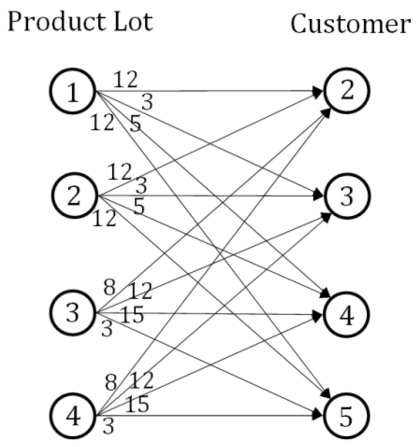

Fig. 3. Simplified assignment model of example case in second round of Algorithm 3.

The illustration of using Algorithm 3 for the same example is expressed as follows. Step 1 of Algorithm 3 returns e_1 = 3 and e_2 = 1. Thus, the problem in the example is simplified based on Algorithm 3 will has the product lots # 1, # 2 and # 3 belong to Supplier 1, and the product lot # 4 belongs to Supplier 2. Then it is noticed that the total number of product lots, i.e. 3 + 1 = 4, is less than the number of all customers, i.e. 5, the simplified problem is infeasible since one customer will not receive a product lot. To repair it, Customer 1 is paired to Supplier 1. Then, ignore Customer 1 from the model and reduce the capacity of Supplier 1 by the demand of Customer 1, and repeat Algorithm 3 from Step 1. The second round of Step 1 of Algorithm 3 then returns e_1 = 2 and e_2 = 2. The simplified problem after ignoring the paired Customer 1 is shown in Fig. 3. The product lots # 1 and # 2 belong to Supplier 1, while the product lots # 3 and # 4 belong to Supplier 2. Now, the number of product lots is not less than to the number of customers, the simplified problem is then feasible as every customer can receive a product lot. The Hungarian method is then used and it returns the same solution to that from Algorithm 2. It is to deliver a product lot from Supplier 1 to Customer 3, a product lot from Supplier 1 to Customer 4, a product lot from Supplier 2 to Customer 2, a product lot from Supplier 2 to Customer 5, and a product lot is delivered from Supplier 1 to Customer 1. Thus the solution in the original problem is to transport three units from Supplier 1 to Customer 3, five units from Supplier 1 to Customer 4, four units from Supplier 2 to Customer 2, three units from Supplier 2 to Customer 5, and nine units from Supplier 1 to Customer 1 with the total costs of \$37. Note that Algorithm 2 and Algorithm 3 will always return the same solution because they receive the same simplified problem which can be formulated into two different models, i.e. transportation model and assignment model.

4. Experiment and Results

This section compares the solution values given by Algorithm 3 to the optimal solution values of the five instances given in Appendix B of [8]. The data of the five instances, i.e. Instances 1 through 5, are also provided to public online in [19]. Instances 1 through 3 each consist of five suppliers and 50 customers, Instance 4 consists of 12 suppliers and 50 customers, and Instance 5 consists of 25 suppliers and 50 customers. Instances 1 and 5 have the high different rate between the demands of customers since the lowest customer's demand is about a half of the highest customer's demand. Instances 2 and 4 have the moderate different rate between the customers' demands since the lowest customer's demand is ¾ of the highest customer's demand. Instance 3 has the low different rate between the customer's demands since the lowest demand is around 95% of the highest demand. Instance 3 is also the simplest instance of these instances because of the low differences between the customer demands. Remind that if the customer demands are all not different, the proposed algorithms both guarantee optimality. Thus, lower different rate between the demands, higher chance to find an optimal solution.

In the experiment, Algorithm 3 uses the Hungarian method coded by a C# program taken from [18] and is run under MS Windows XP on a 2.30 GHz AMD Athlon(tm) 64×2 Dual CPU. The algorithm is run just once per instance because the algorithm is a deterministic algorithm which will always return the same output for a particular input. Note that the solution values from Algorithm 2 and the results from Algorithm 3 are always the same. Table 4 compares, for each instance, the solution value given by Algorithm 3 to the optimal solution taken from [8]. In the table, *m* and *n* are the number of suppliers and the number of customers; Demand Deviation means the percentage of the deviation between the highest demand and the lowest demand of the given customers; Solution Deviation means the percentage of the deviation between the solution value found by the proposed algorithm and the optimal solution value; CPU Time means the computational time consumed by the computer in seconds.

Table 4. Experimental Results

Instance	$m \times n$	Demand Deviation	Optimal Solution	Algorithm Solution	Solution Deviation	CPU Time (sec.)
Instance 1	5 × 50	50%	31130	31554	1.36%	0.06
Instance 2	5 × 50	25%	30378	30378	0.00%	0.11
Instance 3	5 × 50	5%	30297	30297	0.00%	0.11
Instance 4	12 × 50	25%	60074	60074	0.00%	0.08
Instance 5	25 × 50	52%	81612	82967	1.66%	0.03

The results from Table 4 show that, for each instance, the algorithm returns a very good solution within a very short computational time. Instances 1 and 5 seem to be the most difficult instances here because of the highest deviations among the customer demands. The algorithm returns the solution values that higher than the optimal solution values about 1.36% in Instance 1 and 1.66% in Instance 5. Instances 2 through 4 are simpler, since these instances have lower deviation among the customer demands; the algorithm returns the optimal solutions for these three instances. This paper then do the matched-pairs *t*-test to test the hypothesis of H0: the population mean of the solution deviations is equal to or greater than 1.5% versus H1: the population mean of the solution deviations taken from the algorithm is less than 1.5% by using the significance level of 0.05. Since the Minitab software returns the *p*-value of 0.037 which is less than 0.05, the null hypothesis is rejected. It concludes that the population mean of the deviations between the solution values found by the proposed algorithm and the optimal solution values is less than 1.5% with the significance level of 0.05. However, if the given instances are classified into two classes based on the demand deviations, Instance 1 and Instance 5 will belong to high-demand-deviation instances, while Instances 2, 3 and 4 will belong to low-demand-deviation instances. The result of the same hypothesis test on only the high-demand-deviation instances, including Instances 1 and 5, is to fail a rejection of the null hypothesis since the *p*-value is 0.521. It means the population mean of the deviations between the solution values from the algorithms and the optimal solution values is equal to or greater than 1.5% for the high-demand-deviation instances with the significance level of 0.05. But if the hypothesis test is to check whether H0: the population mean of the solution deviations is equal to or greater than 2.5% or H1: the population mean of the solution deviations is less than 2.5% for the high-demand-deviation instances with the significance level of 0.05. The *p*-value of this test is 0.048; it then concludes that the population mean of the solution deviations of the high-demand-deviation instances is less than 2.5% with the significance level of 0.05.

5. Conclusion

This paper presents two heuristic algorithms in order to solve the single-source transportation problems. The two algorithms both transform the problem into the transportation problem where each customer has its demand of one single product lot. The first algorithm then solves the transformed problem using the transportation algorithm while the second algorithm solves the transformed problem using the assignment algorithm. The results returned from the five instances are very good. Based on the results given, the population mean of the deviations between the algorithm's solution values and the optimal solution values is less than 1.5% with the significance level of 0.05.

References

[1] Pentico, D. W. (2007). Assignment problems: A golden anniversary survey. *European Journal of Operational Research*, *176*, 774-793.

[2] Maio, A., & Roveda, C. (1971). An all zero-one algorithm for a certain class of transportation problems. *Operations Research*, *19(6)*, 1406-1418.

[3] Ross, G. T., & Soland, R. M. (1975). A branch-and-bound algorithm for the generalized assignment problem. *Mathematical Science*, *24(3)*, 91-103.

[4] Oncan, T. (2007). A survey of the generalized assignment problem and its applications. *INFOR*, *45(3)*, 123-141.

[5] Srinivasan, V., & Thompson, G. L. (1973). An algorithm for assigning uses to sources in a special class of transportation problems. *Operations Research*, *21(1)*, 284-295.

[6] Nagelhout, R. V., & Thompson, G. L. (1979). *A Single Source Transportation Algorithm*, Technical Report. Carnegie-Mellon University, Pittsburgh.

[7] Srivastava, U. K., Shenoy, G. V., & Sharma, S. C. (1989). *Quantitative Techniques for Managerial Decisions*. New Delhi: New Age International.

[8] Pongchairerks, P. (2002). *An integration between Assignment and Transportation Models*. M.Eng. Thesis, Department of Engineering, Kasetsart University, Thailand.

[9] Pongchairerks, P., & Charnsethikul, P. (2002). An integration between assignment and transportation

models. *Proceedings of Symposium in Production and Quality Engineering for Competitive Business Environment* (pp. 120-130). Thailand, BKK: Kasetsart University.

[10] Pongchairerks, P. (2005). An integration between assignment and transportation models. *Journal of Research in Engineering and Technology*, *2(4)*, 377-390.

[11] Cohen, R., Katzir, L., & Raz, D. (2006). An efficient approximation for the generalized assignment problem. *Information Processing Letters*, *100(4)*, 162-166.

[12] Romeijin, H. E., & Morales, D. R. (2000). A class of greedy algorithms for the generalized assignment problem. *Discrete Applied Mathematics*, *103(1-3)*, 209-235.

[13] Chu, P. C., & Beasley, J. E. (1997). A genetic algorithm for the generalised assignment problem. *Computers and Operations* Research, *24*, 17-23.

[14] Wilson, J. M. (1997). A genetic algorithm for the generalised assignment problem. *Journal of the Operational Research Society*, *48*, 804-809.

[15] Lorena, L. A. N., Narciso, M. G., & Beasley, J. E. (2002). A constructive genetic algorithm for the generalised assignment problem. *Evolutionary Optimization*, *5*, 1-19.

[16] Diaz, J. A. & Fernandez, E. (2001). A tabu search heuristic for the generalized assignment problem. *European Journal of Operational Research*, *132*, 22-38.

[17] Baykasoglu, A., Ozbakir, L., & Tapkan, P. (2007). Artificial bee colony algorithm and its application to generalized assignment problem. In Chan, F. T. S. & Tiwari, M. K. (Eds.), *Swarm Intelligence: Focus on Ant and Particle Swarm Optimization* (pp. 113-144). Vienna: Itech Education and Publishing.

[18] Regueiro, A. (2009). *Hungarian algorithm in C#*. Retrieved June 2, 2014, from Noldorin's Blog: http://blog.noldorin.com/

[19] Pongchairerks, P. (2014). *Single-source transportation problem instances*. Retrieved September 11, 2014, from https://drive.google.com/folderview?id=0B2XqS3TSsvP7SzQ2TFBBeTJJQ3M&usp=sharing.

Pisut Pongchairerks is currently an assistant professor and a head of Production Engineering Program, Faculty of Engineering, Thai-Nichi Institute of Technology, Thailand. He held B.Eng. and M.Eng. degrees in industrial engineering from Industrial Engineering Department, Faculty of Engineering, Kasetsart University and D.Eng. in industrial engineering and management from School of Engineering Technology, Asian Institute of Technology.

He once worked as a teaching assistant in Kasetsart University, and as a senior engineer in Delta Electronics (Thailand). He also served as a lecturer in Sirindhorn International Institute of Technology, and as a senior researcher in M-Focus Company. His currently research interests include operations research, production planning and control, scheduling and sequencing, meta-heuristic algorithms for optimization problems, logistics and supply chain management, and quality control.