RAIRO-Oper. Res. 57 (2023) 913–926
https://doi.org/10.1051/ro/2022152

RAIRO Operations Research
www.rairo-ro.org

A HYBRID SIMULATED ANNEALING ALGORITHM TO ESTIMATE A BETTER UPPER BOUND OF THE MINIMAL TOTAL COST OF A TRANSPORTATION PROBLEM WITH VARYING DEMANDS AND SUPPLIES

Z. A. M. S. Juman[1], M. A. Hoque[2] and Intesar Al-Mudahka[3,*]

Abstract. Minimizing the total cost of transportation of a homogeneous product from multiple sources to multiple destinations when demand at each source and supply at each destination are deterministic and constant is commonly addressed in the literature. However, in practice, demand and supply may fluctuate within a certain range due to variations of the global economy. Subsequently, finding the upper bound of the minimal total cost of this transportation problem with varying demands and supplies (TPVDS) is NP hard. Yet, bounding the minimal total cost is of prime importance for financial sustainability. Although the lower bound of the minimal total cost can be methodologically attained, determining the exact upper bound is challenging. Herein, we demonstrate that existing methods may underestimate this upper minimal total cost bound. We therefore propose an alternative efficient and robust method that is based on the hybridization of simulated annealing and steepest descent. We provide theoretical evidence of its good performance in terms of solution quality and prove its superiority in comparison to existing techniques. We further validate its performance on benchmark and newly generated instances. Finally, we exemplify its utility on a real-world TPVDS.

Mathematics Subject Classification. 90B05, 90B06, 90C08.

Received May 14, 2022. Accepted August 28, 2022.

1. Introduction

Minimizing the total cost of transporting a homogeneous product from multiple sources (*e.g.*, warehouses) to multiple destinations (*e.g.*, supermarkets) subject to supply availability and demand satisfaction is a well-established transportation problem (TP). When the supplies and demands are known and deterministic, TP is solved using a special type of simplex method, warm started from feasible initial solutions that are very easy to construct (*e.g.*, obtained *via* Vogel's approximation method [6]) or are more sophisticated but efficient [5]. However, in practice, the demand and supply of a product may vary, albeit within a certain range, in response to global economic or political events. By simply incorporating interrelated inventory costs incurred during transportation and at destinations into the unitary transport cost, TP oversimplifies the real-life problem. In fact, the inventory costs do also vary within a certain range. In such circumstances, transporters, clients, and

Keywords. Heuristic, upper minimal total cost bound, NP hard problem.

[1] Department of Mathematics Faculty of Science University of Peradeniya, Peradeniya 20400, Sri Lanka.

[2] BRAC Business School, BRAC University, Dhaka 1212, Bangladesh.

[3] Department of Statistics and Operations Research, College of Science, Kuwait University, Kuwait City, Kuwait.

*Corresponding author: intesar.m@ku.edu.kw

suppliers are not merely interested in a deterministic estimate of the total transport cost but are keen in assessing the risks they may be incurring. That is, they wish to bind the minimal total cost so that they better plan their return on investment.

Despite its practical relevance, the TP with non-deterministic supply, demand and unit transport cost has received little attention [1,11]. Kaur and Kumar [8] approximately solved a TP with fixed demands and supplies but uncertain imprecise values of transportation cost. Their algorithm represented the transportation costs by generalized trapezoidal fuzzy numbers. Its complexity was later further enhanced [2]. Liu *et al.* [10] approximated the minimal transport cost of solids using a fuzzy simulation-based tabu search. Liu [9] proposed a heuristic H_{L} that calculates the minimal total cost bounds $\underline{z}_{\mathrm{L}}$ and $\overline{z}_{\mathrm{L}}$ of the transportation problem with demand and supply quantities varying within their respective ranges using a pair of mathematical programs. Juman and Hoque [4] demonstrated that H_{L} may, in some instances, fail to identify a correct upper minimal total cost bound. They further extended the transportation problem to include the interrelated inventory costs during transportation and at destinations. They then developed heuristic H_{JH} that finds a lower bound $\underline{z}_{\mathrm{JH}}$ and an upper bound $\overline{z}_{\mathrm{JH}}$ to the minimal total cost of their extended model. For all tested instances, $\underline{z}_{\mathrm{JH}} = \underline{z}_{\mathrm{L}}$. For small and medium sized instances, $\overline{z}_{\mathrm{JH}} \leq \underline{z}_{\mathrm{L}}$. This advantage did not however extend to large sized instances.

In this paper, we consider a TP with varying demands and supplies including inventories (TPVDSII) where the objective is to minimize total cost. Because the supply and demand vary within a predefined range, the problem reduces to estimating the lower bound $\underline{z}$ and upper bound $\overline{z}$ of the minimal total cost. Although $\underline{z}$ can be found methodologically, determining $\overline{z}$ is challenging: It involves an NP-hard problem. Estimating $\overline{z}$ heuristically is possible but may have short comings as in the cases of $\overline{z}_{\mathrm{L}}$ and $\overline{z}_{\mathrm{JH}}$. In this paper, we overcome this shortcoming of worst-case performance by tightening $\overline{z}_{\mathrm{JH}}$. We propose a new efficient and robust method that either matches or improves existing upper bounds. It incorporates more combinatorial choices of supply and demand than H_{JH} while it avoids the exhaustive enumeration of the $(m+n)!\prod_{i=1}^{m}(\Delta s_i+1)\prod_{j=1}^{n}(\Delta d_j+1)$ potential combinations of choices of supplies and demands within their respective ranges of the TPVDSII, where m and n are the numbers of suppliers and buyers respectively, Δs_i is the range of the supply quantity of the ith supplier, and Δd_j is the range of the demand quantity of the jth buyer. In a real-life yoghurts' distribution problem [7], the number of combinations is 1.678110^{157}. Enumerating all these supply-demand scenarios to estimate the exact upper bound cost is impractical. So, the development of a streamline method to tackle these choices of supply-demand would be challenging. Our new approach, labelled hereafter H, incorporates more combinatorial choices of supply and demand than H_{JH}.

This paper addresses a pertinent real-life supply chain logistics' problem that occurs in many circumstances, such as the distribution of yoghurt, tea, raw material, etc. It proves that existing methods H_{L} and H_{JH} underestimate the upper bound $\overline{z}$, and propose a better near-optimal estimation of the worst-case realization on the least aggregated expenses of the TPVDSII. It applies the estimation technique H to a real-world yoghurt show case. It further undertakes an extensive numerical experiment using benchmark and newly generated random instances to assess the performance of H.

Section 2 describes the problem. Section 3 details H_{JH} and illustrates its shortcoming. Section 4 details the new efficient robust method H that finds a tighter upper minimal total cost bound. Section 5 compares the performance of H to existing methods. Section 6 concludes the paper and gives potential extensions.

2. Problem definition

TPVDSII is a transportation problem such that the following assumptions apply.

(1) A homogeneous product is transported from m suppliers to n buyers.
(2) The demand $\hat{d}_j$ of buyer $j,\ j=1,\ldots,n$, for the product is deterministic and integer but variable over time within a bounded interval:

$$\hat{d}_j \in \left[\underline{D}_j, \overline{D}_j\right].$$

Consequently, total demand is integer and varies over time in a bounded interval:

$$\sum_{j=1}^{n} \hat{d}_j \in \left[\sum_{j=1}^{n} \underline{D}_j, \sum_{j=1}^{n} \overline{D}_j\right].$$

(3) The supply $\hat{s}_i$ of supplier i, $i = 1, \ldots, m$, for the product is deterministic and integer but variable over time within the a bounded interval:

$$\hat{s}_i \in \left[\underline{S}_i, \overline{S}_i\right].$$

Consequently, the total supply is integer and varies over time in a bounded interval:

$$\sum_{i=1}^{m} \hat{s}_i \in \left[\sum_{i=1}^{m} \underline{S}_i, \sum_{i=1}^{m} \overline{S}_i\right].$$

(4) Total supply equals or exceeds total demand; *i.e.*,

$$\sum_{i=1}^{m} \hat{s}_i \geq \sum_{j=1}^{n} \hat{d}_j.$$

(5) Each buyer j, $j = 1, \ldots, n$, has enough storage capacity to accommodate the required inventory. The unit inventory holding cost per period at buyer j, $j = 1, \ldots, n$, is h_j.
(6) A transport vehicle is available to transport the required shipment quantities. The unit transportation cost and time of delivery from a supplier i, $i = 1, \ldots, m$, to a buyer j, $j = 1, \ldots, n$, are c_{ij} and t_{ij}, respectively.
(7) The quantity x_{ij} shipped from a supplier i, $i = 1, \ldots, m$, to a buyer j, $j = 1, \ldots, n$, reaches j within the same period.

Using the above notation and assumptions, we model TPVDSII using the integer decision variables x_{ij}, which denote the quantities shipped from supplier i, $i = 1, \ldots, m$, to buyer j, $j = 1, \ldots, n$.

$$\min z = \sum_{i=1}^{m} \sum_{j=1}^{n} a_{ij} x_{ij} \tag{1}$$

$$\text{s.t.} \sum_{j=1}^{n} x_{ij} \leq \hat{s}_i \qquad i = 1, \ldots, m \tag{2}$$

$$\sum_{i=1}^{m} x_{ij} \leq \hat{d}_j \qquad j = 1, \ldots, n \tag{3}$$

$$x_{ij} \geq 0 \qquad i = 1, \ldots, m, j = 1, \ldots, n, \tag{4}$$

where $a_{ij} = c_{ij} + \frac{1}{2} h_j + t_{ij} h_j$. The model (1)–(4) corresponds to an integer linear program. It has at least one feasible solution if the sum of supply quantities exceeds the sum of demand quantities:

$$\sum_{i=1}^{m} \hat{s}_i \geq \sum_{j=1}^{n} \hat{d}_j.$$

It can be solved as a linear program with at least one of its optimal solutions being integer when $\hat{s}_i$, $i = 1, \ldots, m$, $\hat{d}_j$, $j = 1, \ldots, n$, and cost coefficients a_{ij} are integer. In such a case, the model (1)–(4) can be easily solved using the simplex method for transportation problems.

To obtain an upper bound $\overline{z}$ to the minimal total cost z, we treat the supply $\hat{s}_i$, $i = 1, \ldots, m$, and demand $\hat{d}_j$, $j = 1, \ldots, n$, as decision variables that vary within their respective ranges. $\overline{z}$ is therefore the solution value of the linear program

$$\begin{aligned} &\text{Equation (1)–(4)} \\ \text{s.t. } &\hat{s}_i \in \left[\underline{S}_i, \overline{S}_i\right] \qquad i = 1, \ldots, m \qquad (5) \\ &\hat{d}_j \in \left[\underline{D}_j, \overline{D}_j\right] \qquad j = 1, \ldots, n. \qquad (6) \end{aligned}$$

For small-sized instances and ranges of supply and demand, the model (1)–(6) can be solved using a linear programming off-the-shelf solver such as LINGO. However, this becomes challenging as the problem size increases.

3. Existing upper bound

$H_{\rm JH}$ is a partial enumeration heuristic that calculates $\overline{z}_{\rm JH}$, an estimate of the upper bound to the minimal total cost of model (1)–(6); *i.e.*, when $\hat{s}_i,\ i = 1, \ldots, m$, and $\hat{d}_j,\ j = 1, \ldots, n$, are variable over time. $H_{\rm JH}$ sets $\hat{s}_i,\ i = 1, \ldots, m$, and $\hat{d}_j,\ j = 1, \ldots, n$, to their respective upper bound values $\overline{S}_i$ and $\overline{D}_j$, and solves model (1)–(4) to obtain total cost z_0. Subsequently, $H_{\rm JH}$ initializes the upper bound $\overline{z}_J H$ of the minimal total cost z to z_0. For all possible pairs of (i, j) supplier buyer, at each iteration, $H_{\rm JH}$ reduces $\hat{s}_i$ and $\hat{d}_j$ by 1, computes the resulting minimal cost z_{ij} by solving model (1)–(4), and updates the upper bound $\overline{z}_{\rm JH}$ if $z_{ij} > z_0$; in fact, $\overline{z}_{\rm JH} = \max\{z_0, z_{ij}\}$. Let $(\mathbf{x}, \hat{\mathbf{s}}, \hat{\mathbf{d}})_{\rm JH} = (\mathbf{x}_{\rm JH}, \hat{\mathbf{s}}_{\rm JH}, \hat{\mathbf{d}}_{\rm JH})$ be the resulting solution and $\overline{z}_{\rm JH}$ the corresponding solution value.

The application of $H_{\rm JH}$ to the large-scale problem 4 of Table 3 in Juman and Hoque [4] yields $\overline{z}_{\rm JH} = 3840$, corresponding to the solution $\mathbf{x}_{\rm JH}$ whose non-zero entries are: $x_{12} = 20,\ x_{21} = 40,\ x_{23} = 40,\ x_{210} = 20,\ x_{46} = 20, x_{49} = 120,\ x_{54} = 80,\ x_{78} = 60,\ x_{85} = 40,\ x_{87} = 40,\ x_{93} = 20$. $\mathbf{x}_{\rm JH}$ is obtained when supply $\hat{\mathbf{s}}_{\rm JH} = [200; 250; 300; 150; 400; 200; 250; 300; 150; 400]$ and demand $\hat{\mathbf{d}}_{\rm JH} = [100; 125; 150; 75; 200; 100; 125; 150; 75; 200]$. However, $\overline{z}_{\rm JH}$ underestimates the true upper bound $\overline{z}$. Solving the model (1)–(6) using LINGO, with no preset run time, yields a minimal total transportation cost $\overline{z} = 4240$, corresponding to $(\mathbf{x}, \hat{\mathbf{s}}, \hat{\mathbf{d}})$ where the non-zero entries of $\mathbf{x}$ are: $x_{12} = 20,\ x_{21} = 40,\ x_{23} = 20,\ x_{29} = 45,\ x_{2,10} = 20,\ x_{36} = 20,\ x_{49} = 75,\ x_{54} = 80,\ x_{78} = 60, x_{85} = 40,\ x_{87} = 40,\ x_{93} = 40$, the supply $\hat{\mathbf{s}} = [100; 125; 150; 75; 200; 100; 125; 150; 75; 200]$, and demand $\hat{\mathbf{s}} = [40; 20; 60; 80; 40; 20; 40; 60; 120; 20]$. It follows that $\overline{z} = 4240 > 3840 = \overline{z}_{\rm JH}$. Thus, $\overline{z}_{\rm JH}$ underestimates the true $\overline{z}$ (Tab. 1).

This further occurs in each of the 2-supplier 3-buyer examples of Table 2, where the lower and upper bounds of the demand and supply quantities are $\underline{\mathbf{D}} = \left(45\ 30\ 60\right)$, $\overline{\mathbf{D}} = \left(90\ 60\ 120\right)$, $\underline{\mathbf{S}} = \left(60\ 75\right)$, $\overline{\mathbf{S}} = \left(120\ 150\right)$. When the problem size is large, LINGO neither converges to the optimum $\overline{z}$ nor proves its optimality. For those TPVDS instances, an efficient and robust heuristic to find a tighter upper minimal total cost bound is a viable alternative.

4. An efficient robust heuristic to estimate the upper minimal total cost bound

The proposed heuristic H starts from the solution $(\mathbf{x}, \hat{\mathbf{s}}, \hat{\mathbf{d}})_0 = (\mathbf{x}, \hat{\mathbf{s}}, \hat{\mathbf{d}})_{\rm JH}$ and initial solution value $z_0 = z_{\rm JH}$. Its iterative steps proceed depending on which of the three cases i–iii occurs.

Case i

When $\sum_{i=1}^{m} \hat{s}_i = \sum_{j=1}^{n} \hat{d}_j$ and $z_0 = z_{\rm JH}$, H considers two actions. First, it keeps all $s_i,\ i = 1, \ldots, m$, fixed but increases one d_j value and decreases another $d_{j'}$ by the same amount such that the equality $\sum_{i=1}^{m} \hat{s}_i = \sum_{j=1}^{n} \hat{d}_j$ still holds. Second, it keeps all $d_j,\ j = 1, \ldots, n$, fixed but increases one s_i value and decreases another $s_{i'}$ value by the same amount so that the equality $\sum_{i=1}^{m} \hat{s}_i = \sum_{j=1}^{n} \hat{d}_j$ still holds. Either way, the resulting feasible solutions may have a minimal total cost greater than z_0. Algorithms 1 and 2 address each of these two actions, respectively.

Algorithm 1 considers consecutive pairs (j, j') of buyers. Starting from an initial feasible solution $(\mathbf{x}, \hat{\mathbf{s}}, \hat{\mathbf{d}})_0$, it initializes $j = 1,\ j' = 2$, and $z_{jj'} = 0$. Successively, it reduces $\hat{d}_j$ by 1 and increases $\hat{d}_{j'}$ by 1 while keeping $\hat{s}_i \leq \overline{S}_i,\ i, i = 1, \ldots, m$, and the remaining $\hat{d}_j$ values fixed at their initial values. For each pair (j, j'), it calculates

TABLE 1. Three example where H_{JH} underestimates $\overline{z}$.

#	1	2	3
$\mathbf{c}$	$\begin{pmatrix}15 & 10 & 20\\ 15 & 10 & 40\end{pmatrix}$	$\begin{pmatrix}30 & 40 & 20\\ 15 & 10 & 90\end{pmatrix}$	$\begin{pmatrix}11 & 25 & 45\\ 115 & 25 & 40\end{pmatrix}$
$\mathbf{x}_{\text{JH}}$	$\begin{pmatrix}0 & 0 & 45\\ 45 & 45 & 60\end{pmatrix}$	$\begin{pmatrix}0 & 0 & 60\\ 45 & 45 & 60\end{pmatrix}$	$\begin{pmatrix}60 & 0 & 0\\ 30 & 60 & 60\end{pmatrix}$
$\hat{\mathbf{s}}_{\text{JH}}$	(60 150)	(60 150)	(60 150)
$\hat{\mathbf{d}}_{\text{JH}}$	(45 45 120)	(45 45 120)	(90 60 150)
$\overline{z}_{\text{JH}}$	4725	7725	8010
$\mathbf{x}$	$\begin{pmatrix}0 & 0 & 60\\ 60 & 30 & 60\end{pmatrix}$	$\begin{pmatrix}0 & 0 & 60\\ 60 & 30 & 60\end{pmatrix}$	$\begin{pmatrix}60 & 0 & 0\\ 30 & 30 & 90\end{pmatrix}$
$\hat{\mathbf{s}}$	(60 150)	(60 150)	(60 150)
$\hat{\mathbf{d}}$	(60 30 120)	(60 30 120)	(90 30 90)
$\overline{z}$	4800	7800	8460

the minimal total cost $z_{jj'}(\hat{d}_j, \hat{d}_{j'})$. It updates $\overline{z}_{jj'}$ setting it to $\max\{\overline{z}_{jj'}, z_{jj'}(\hat{d}_j, \hat{d}_{j'})\}$ and retains the values $(\hat{d}_j, \hat{d}_{j'})$ that correspond to $\overline{z}_{jj'}$. It continues its iterations until either $\hat{d}_j = \underline{D}_j$ or $\hat{d}_{j'} = \overline{D}_{j'}$. If $\overline{z}_{jj'} > \overline{z}$, then Algorithm 1 updates the upper bound $\overline{z}$ setting it to $\overline{z}_{jj'}$. Algorithm 1 then iterates over all possible (j, j') combinations. It stops when it has considered all pairs (j, j') of buyers. The updated $\overline{z}$ value will be the upper minimal total cost bound.

Algorithm 2 considers consecutive pairs (i, i') of vendors. Starting from an initial feasible solution $(\mathbf{x}, \hat{\mathbf{s}}, \hat{\mathbf{d}})_0$, it initializes $i = 1$, $i' = 2$, and $\overline{z}_{ii'} = 0$. Successively, it reduces $\hat{s}_i$ by 1 and increases $\hat{s}_{i'}$ by 1 while keeping $\hat{d}_j \leq \overline{D}_j$, $j = 1, \ldots, n$, and the remaining $\hat{s}_i$ values fixed at their initial values. For each pair (i, i'), it calculates the minimal total cost $z_{ii'}(\hat{s}_i, \hat{s}_{i'})$. It updates $z_{ii'}$ to $\max\{\overline{z}_{ii'}, z_{ii'}(\hat{s}_i, \hat{s}_{i'})\}$ and retains the values $(\hat{s}_i, \hat{s}_{i'})$ that correspond to $\overline{z}_{ii'}$. It continues its iterations until either $\hat{s}_i = \underline{S}_i$ or $\hat{s}_{i'} = \overline{S}_{j'}$. If $\overline{z}_{ii'} > \overline{z}$, then Algorithm 2 updates the upper bound $\overline{z}$ setting it to $\overline{z}_{ii'}$. Subsequently, it iterates over all possible (i, i') combinations. It stops when it has considered all pairs (i,i') of vendors. The updated $\overline{z}$ value will be the upper minimal total cost bound.

Case ii

When $\sum_{i=1}^{m} \hat{s}_i < \sum_{j=1}^{n} \hat{d}_j$ and $z_0 = z_{\text{JH}}$, H undertakes three actions, all of which maintain the inequality. The first keeps all s_i, $i = 1, \ldots, m$, fixed but increases one d_j value and decreases another $d_{j'}$ by the same amount. The second keeps all d_j, $j = 1, \ldots, n$, fixed but increases one s_i value and decreases another $s_{i'}$ value by the same amount. The third keeps all d_j, $j = 1, \ldots, n$, fixed but decreases one s_i value by one. Any of these actions may generate feasible solutions whose minimal total cost greater than z_0. Algorithms 1–3 address each of these respective actions.

Algorithm 3 considers each vendor i, $i = 1, \ldots, m$. Starting from an initial feasible solution $(\mathbf{x}, \hat{\mathbf{s}}, \hat{\mathbf{d}})_0$, it initializes $i = 1$, and $\overline{z}_i = 0$. Successively, it reduces $\hat{s}_i$ by 1 while keeping $\hat{d}_j \leq \overline{D}_j$, $j = 1, \ldots, n$, and the remaining $\hat{s}_i$ values fixed at their initial values. It calculates the minimal total cost $z_i(\hat{s}_i)$. It updates $\overline{z}_i$ setting it to $\max\{\overline{z}_i, z_i(\hat{s}_i)\}$ and retains the values $\hat{s}_i^N$ that correspond to $\overline{z}_i$. It continues its iterations until either $\hat{s}_i = \underline{S}_i$ or $\hat{s}_i = \overline{S}_i$. If $\overline{z}_i > \overline{z}$, then Algorithm 3 updates the upper bound $\overline{z}$ setting it to $\overline{z}_i$. It then iterates over all possible vendors.

Once it has considered all possible vendors, Algorithm 3 considers every buyer j, $j = 1, \ldots, n$. It initializes $j = 1$ and successively reduces $\hat{d}_j$ by 1 while keeping $\hat{d}_j \leq \overline{D}_j$, and the remaining $\hat{d}_j$ values fixed at their initial

Algorithm 1. Iterating over demand for case i.

1: **Input:** An initial solution $(\mathbf{x}, \hat{\mathbf{s}}, \hat{\mathbf{d}})_0$
2: Set $j = 1,\ j' = 1$ and $\overline{z}_{jj'} = 0$.
3: **if** $j = j'$ **then**
4: Set $j' = j' + 1$.
5: **end if**
6: $N = 1$.
7: **if** $\hat{d}_j \neq \underline{D}_j$ and $\hat{d}_{j'} \neq \overline{D}_j$ **then**
8: Decrease $\hat{d}_j$ by 1 and increase $\hat{d}_{j'}$ by 1.
9: Solve model (1)–(4) to get $z_{jj'}(\hat{d}_j, \hat{d}_{j'})$ and $x^N_{jj'}$.
10: **if** $\overline{z}_{jj'} > z_{jj'}(\hat{d}_j, \hat{d}_{j'})$ **then**
11: Set $\overline{z}_{jj'} = z_{jj'}(\hat{d}_j, \hat{d}_{j'}),\ x_{jj'} = x^N_{jj'},\ \hat{\mathbf{d}}^N = \hat{\mathbf{d}}'$ with $\hat{d}^N_j = \hat{d}_j,\ \hat{d}^N_{j'} = \hat{d}_{j'}$.
12: **end if**
13: **else if** $\hat{d}_j = D_j$ **then**
14: Go to Line 28.
15: **else if** $\hat{d}_{j'} = \overline{D}_{j'}$ **then**
16: Go to Line 25.
17: **end if**
18: **if** $N < \min\{\overline{D}_j - \underline{D}_j, \overline{D}_{j'} - \underline{D}_{j'}\}$ **then**
19: Set $N = N + 1$.
20: Go to Line 6.
21: **end if**
22: **if** $\overline{z}_{jj'} > \overline{z}$ **then**,
23: Set $\overline{z} = \overline{z}_{jj'}$ and $(\mathbf{x}, \hat{\mathbf{s}}, \hat{\mathbf{d}}) = \left((x^N_{jj'}), \hat{\mathbf{s}}, \hat{\mathbf{d}}^N\right)$.
24: **end if**
25: **if** $j < n$, **then**
26: Set $j = j + 1$. Go to Line 3.
27: **end if**
28: **if** $j' < n$, **then**
29: Set $j' = j' + 1$. Go to Line 3.
30: **end if**
31: **Output:** $\overline{z}$ and $(\mathbf{x}, \hat{\mathbf{s}}, \hat{\mathbf{d}})$

values. For each j, it calculates the minimal total cost $z_j(\hat{d}_j)$. It updates $\overline{z}_j$ to $\max\{\overline{z}_j, z_j(\hat{d}_j)\}$ and retains the values $\hat{d}^N_j$ that correspond to $\overline{z}_j$. It continues its iterations until either $\hat{d}_j = \underline{D}_j$ or $\hat{d}_j = \overline{D}_j$. If $\overline{z}_j > \overline{z}$, Algorithm 3 updates the upper bound $\overline{z}$ setting it to $\overline{z}_j$. It then iterates over all possible j before it stops. The updated $\overline{z}$ value will be the upper minimal total cost bound.

Case iii

When $\sum_{i=1}^{m} \hat{s}_i = \sum_{j=1}^{n} \hat{d}_j$ and $z_0 \neq z_{\text{JH}}$, the current z_0 value is an upper minimal total cost bound.

Proposed approach

Our efficient and robust method H addresses the above three cases, using Algorithm 4, which is a steepest descent (SD) that estimates an upper bound to the minimal total cost bound.

The following theorem highlights the good quality of the upper bound $\overline{z}$ in comparison to $\overline{z}_{\text{JH}}$.

Theorem 4.1. *Algorithm 4 yields an upper bound estimate $\overline{z}$ that is at least as good as or better than $\overline{z}_{\text{JH}}$.*

Proof. Algorithm 4 uses H_{JH} to generate its initial solution $(\mathbf{x}, \hat{\mathbf{s}}, \hat{\mathbf{d}})_0$ and initial bound $\overline{z}_0$; in fact, it sets $\overline{z} = z_0$ and $(\mathbf{x}, \hat{\mathbf{s}}, \hat{\mathbf{d}}) = (\mathbf{x}, \hat{\mathbf{s}}, \hat{\mathbf{d}})_0$, where $(\mathbf{x}, \hat{\mathbf{s}}, \hat{\mathbf{d}})_0$ and z_0 are the output of H_{JH}. At each iteration, it generates new solutions by varying the supply and demand within their respective ranges, calculating the resulting total cost and updating $\overline{z}$ whenever a higher cost solution value is encountered. Thus, $z \geq z_0$. □

Algorithm 4 searches for an upper bound of the total transport and inventory cost. It improves existing bounds of benchmark instances. Yet, it is an iterative SD; thus, has a limited capability of diversifying the

Algorithm 2. Iterating over supply for case i.

1: **Input:** An initial solution $(\mathbf{x}, \hat{\mathbf{s}}, \hat{\mathbf{d}})_0$
2: Set $i = 1,\ i' = 1$ and $\overline{z}_{ii'} = 0$.
3: **if** $i = i'$ **then**
4: Set $i' = i' + 1$.
5: **end if**
6: $N = 1$.
7: **if** $\hat{s}_i \neq \underline{S}_i$ and $\hat{s}_{i'} \neq \overline{S}_i$ **then**
8: Decrease $\hat{s}_i$ by 1 and increase $\hat{s}_{i'}$ by 1.
9: Solve model (1)–(4) to get $z_{ii'}(\hat{s}_i, \hat{s}_{i'})$ and $x^N_{ii'}$.
10: **if** $\overline{z}_{ii'} > z_{ii'}(\hat{s}_i, \hat{s}_{i'})$ **then**
11: Set $\overline{z}_{ii'} = z_{ii'}(\hat{s}_i, \hat{s}_{i'}),\ x_{ii'} = x^N_{ii'},\ \hat{\mathbf{s}}^N = \hat{\mathbf{s}}'$ with $\hat{s}^N_i = \hat{s}_i,\ \hat{s}^N_{i'} = \hat{s}_{i'}$.
12: **end if**
13: **else if** $\hat{s}_i = \underline{S}_i$ **then**
14: Go to Line 28.
15: **else if** $\hat{s}_{i'} = \overline{S}_{i'}$ **then**
16: Go to Line 25.
17: **end if**
18: **if** $N < \min\{\overline{S}_i - \underline{S}_i, \overline{S}_{i'} - \underline{S}_{i'}\}$ **then**
19: Set $N = N + 1$.
20: Go to Line 6.
21: **end if**
22: **if** $\overline{z}_{ii'} > \overline{z}$ **then**,
23: Set $\overline{z} = \overline{z}_{ii'}$ and $(\mathbf{x}, \hat{\mathbf{s}}, \hat{\mathbf{d}}) = \left((x^N_{ii'}), \hat{\mathbf{s}}^N, \hat{\mathbf{d}}\right)$.
24: **end if**
25: **if** $j < n$, **then**
26: Set $i = i + 1$. Go to Line 3.
27: **end if**
28: **if** $i' < n$, **then**
29: Set $i' = i' + 1$. Go to Line 3.
30: **end if**
31: **Output:** $\overline{z}$ and $(\mathbf{x}, \hat{\mathbf{s}}, \hat{\mathbf{d}})$

search. As such, it is coupled with a stochastic exploration that allows it to mimic the behaviour of a simulated annealing (SA). Specifically, we generate solutions using SD and subject them to SA, which estimates the upper minimal total cost bound of the TPVDSII. Algorithm 5 details the steps of the proposed approach, labelled H.

5. Computational results

We compare our proposed approach H against H_{L} and H_{JH} for both small and large size instances, and for a real-world application. The methods are coded in MATLAB and run on a personal computer with Intel Core (TM) 2 Duo CPU T7700, 2.40 GHz, and RAM 1.99 GB.

5.1. Small sized problems

First, we compare $\overline{z}$ to $\overline{z}_{\mathrm{JH}}$ and $\overline{z}_{\mathrm{L}}$ on fifteen small sized numerical problems: problems 1–7 are randomly generated new instances whereas problems 8–15 are benchmark instances [4]. Data for problems 1–5 are given in Table A.1. Data for the 2-supplier 3-buyer problem 6 follows: $\underline{D}_1 = 45,\ \overline{D}_1 = 90; \underline{D}_2 = 30,\ \overline{D}_2 = 60; \underline{D}_3 = 60,\ \overline{D}_1 = 120; \underline{S}_1 = 60,\ \overline{S}_1 = 120; \underline{S}_2 = 75,\ \overline{S}_2 = 150; c_{11} = 11,\ c_{12} = 25,\ c_{13} = 4;\ c_{21} = 15,\ c_{22} = 5,\ c_{23} = 40$. The numerical problem 7 is generated by replacing $c_{11} = 30$ by $c_{11} = 130$ in Problem 13.

Table 3 compares (i) the bounds $\overline{z}$ to $\overline{z}_{\mathrm{JH}}$ and $\overline{z}_{\mathrm{L}}$ and (ii) the run time RT needed to calculate $\overline{z}$ to the run times $\mathrm{RT_L}$ and $\mathrm{RT_{JH}}$ needed to compute $\overline{z}_{\mathrm{L}}$ and $\overline{z}_{\mathrm{JH}}$, respectively, where all run times are in seconds. The last two columns of Table 3 report the percent improvement Δ_* brought H with respect to heuristic $*$, where $* = H_{\mathrm{L}}, H_{\mathrm{JH}}$ and $\Delta_* = \frac{\overline{z} - \overline{z}_*}{\overline{z}_*} 100\%$.

Table 3 and specifically its last two columns indicate that H improves the upper bounds obtained by H_{L} and H_{JH} by as much as 94.90% and 90.50%, respectively. This improvement occurs for problem 3, where both

Algorithm 3. Iterating over supply for case ii.

1: **Input:** An initial solution $(\mathbf{x}, \hat{\mathbf{s}}, \hat{\mathbf{d}})_0$
2: Set $i = 1,\ N = 1$ and $\overline{z}_i = 0$.
3: **if** $\hat{s}_i \neq \underline{S}_i$ and $\hat{s}_i \neq \overline{S}_i$ **then**
4: Decrease $\hat{s}_i$ by 1.
5: Solve model (1)–(4) to get $z_i(\hat{s}_i)$ and x_i^N.
6: **if** $\overline{z}_i > z_i(\hat{s}_i)$ **then**
7: Set $\overline{z}_i = z_i(\hat{s}_i),\ x_i = x_i^N,\ \hat{\mathbf{s}}^N = \hat{\mathbf{s}}'$ with $\hat{s}_i^N = \hat{s}_i$.
8: **end if**
9: **end if**
10: **if** $N < \min\{\overline{S}_i - \underline{S}_i\}$ and $\sum_{i=1}^m \hat{s}_i > \sum_{j=1}^n \hat{d}_j$ **then**
11: Set $N = N + 1$.
12: Go to Line 10.
13: **end if**
14: **if** $\overline{z}_i > \overline{z}$ **then**,
15: Set $\overline{z} = \overline{z}_i$ and $(\mathbf{x}, \hat{\mathbf{s}}, \hat{\mathbf{d}}) = \Big((x_i^N), \hat{\mathbf{s}}^N, \hat{\mathbf{d}}\Big)$.
16: Go to Line 3
17: **end if**
18: **if** $i < m$, **then**
19: Set $i = i + 1$.
20: Go to Line 3.
21: **end if**
22: **if** $\sum_{i=1}^m \hat{s}_i = \sum_{j=1}^n \hat{d}_j$ **then**
23: Apply Algorithm 1.
24: **else if** $\sum_{i=1}^m \hat{s}_i > \sum_{j=1}^n \hat{d}_j$ **then**
25: Stop.
26: **else**
27: Continue.
28: **end if**
29: Set $j = 1$, and $N = 1$.
30: **if** $\hat{d}_j \neq \underline{D}_j$ and $\hat{d}_j \neq \overline{D}_j$ **then**
31: Decrease $\hat{d}_j$ by 1.
32: Solve model (1)–(4) to get $z_j(\hat{d}_j)$ and x_j^N.
33: **if** $\overline{z}_j > z_j(\hat{d}_j)$ **then**
34: Set $\overline{z}_j = z_j(\hat{d}_j),\ x_j = x_j^N,\ \hat{\mathbf{d}}^N = \hat{\mathbf{d}}'$ with $\hat{d}_j^N = \hat{d}_j$.
35: **end if**
36: **end if**
37: **if** $N < \min\{\overline{D}_j - \underline{D}_j\}$ and $\sum_{i=1}^m \hat{s}_i > \sum_{j=1}^n \hat{d}_j$ **then**
38: Set $N = N + 1$.
39: Go to Line 30.
40: **end if**
41: **if** $\overline{z}_j > \overline{z}$ **then**,
42: Set $\overline{z} = \overline{z}_j$ and $(\mathbf{x}, \hat{\mathbf{s}}, \hat{\mathbf{d}}) = \Big((x_j^N), \hat{\mathbf{s}}^N, \hat{\mathbf{d}}\Big)$.
43: **end if**
44: **if** $j < n$ **then**
45: Set $j = j + 1$.
46: Go to Line 37.
47: **end if**
48: **Output:** $\overline{z}$ and $(\mathbf{x}, \hat{\mathbf{s}}, \hat{\mathbf{d}})$

$\overline{z}_{\mathrm{L}}$ and $\overline{z}_{\mathrm{JH}}$ underestimate $\overline{z}$, most likely because H_{L} and H_{JH} miss the scenarios that cause such high total costs during their partial enumeration. This improvement occurs at a slighter increase of run time. However, the difference, which is of the order of 0.35 s, is negligible compared to the sizeable impact of a better estimation of the upper bound. For all tested instances, LINGO obtains the same solution value as H when allocated 3600 s of run time.

Algorithm 4. Steepest descent (SD).

1: **Input:** A problem instance.
2: Set $i = 1,\ N = 1$ and $\overline{z}_i = 0$.
3: Apply H_{JH} to obtain an initial solution $(\mathbf{x}, \hat{\mathbf{s}}, \hat{\mathbf{d}})_0$ and its optimal solution value z_0.
4: Set $z = z_0$, and $(\mathbf{x}, \hat{\mathbf{s}}, \hat{\mathbf{d}}) = (\mathbf{x}, \hat{\mathbf{s}}, \hat{\mathbf{d}})_0$.
5: **if** $\sum_{i=1}^{m} \hat{s}_i > \sum_{j=1}^{n} \hat{d}_j$ and $z \neq z_0$ **then** Go to Line 9.
6: **else if** $\sum_{i=1}^{m} \hat{s}_i = \sum_{j=1}^{n} \hat{d}_j$ and $z \neq z_0$ **then** Go to Line 10.
7: **else**Stop.
8: **end if**
9: Apply Algorithm 3 to obtain $z(3)$ and $(\mathbf{x}, \hat{\mathbf{s}}, \hat{\mathbf{d}})_{(3)}$.
10: Apply Algorithm 1 to obtain $z(1)$ and $(\mathbf{x}, \hat{\mathbf{s}}, \hat{\mathbf{d}})_{(1)}$.
11: Apply Algorithm 2 to obtain $z(2)$ and $(\mathbf{x}, \hat{\mathbf{s}}, \hat{\mathbf{d}})_{(2)}$.
12: Set $z = \overline{z}_{(k^*)} = \max_{k=1,2,3}\{\overline{z}_{(k)}\}$ and $(\mathbf{x}, \hat{\mathbf{s}}, \hat{\mathbf{d}}) = (\mathbf{x}, \hat{\mathbf{s}}, \hat{\mathbf{d}})_{(k^*)}$.
13: **Output:** $\overline{z}$ and $(\mathbf{x}, \hat{\mathbf{s}}, \hat{\mathbf{d}})$.

Algorithm 5. Heuristic H.

1: **Input:** A problem instance.
2: Set $k = 0$.
3: Apply SD to obtain an initial solution $(\mathbf{x}, \hat{\mathbf{s}}, \hat{\mathbf{d}})^0$ and its optimal solution value $z(\mathbf{x}^k)$.
4: Set the initial temperature $T_0 = 0.9$, the cooling factor $\rho = 0.5$, $\bar{k}$ the length of the SA plateau, and M the number of plateaus.
5: Set $T = T_0$, the best feasible solution $\mathbf{x}^* = \mathbf{x}^0$, and its cost value $z^* = z(\mathbf{x}^0)$.
6: **while** $k \leq \bar{k}$ **do**
7: **while** $i \leq m$ **do**
8: **while** $j \leq n$ **do**
9: Set $s_i^{(k+1)} = \lceil s_i^{(k)} + (\overline{S}_i^{(k)} - \underline{S}_i^{(k)}) r_k \rceil$, where r_k Uniform $(0, 1)$.
10: Set $d_j^{(k+1)} = \lceil d_j^{(k)} + (\overline{D}_j^{(k)} - \underline{D}_j^{(k)}) r_k \rceil$, where r_k Uniform $(0, 1)$.
11: Substitute $s_i^{(k+1)}$ and $d_j^{(k+1)}$ in equations (1)–(4)- and obtain $\mathbf{x}^(k+1)$ and $z(\mathbf{x}^{(k+1)})$.
12: Set $\beta_z = z(\mathbf{x}^{(k+1)}) - z(\mathbf{x}^{(k)})$.
13: **if** $\beta_z \geq 0$ or $e^{\left(-\frac{Beta_z}{T_k}\right)} > r$, where r Uniform $(0, 1)$. **then**
14: Update $\mathbf{x}^* = \mathbf{x}^{(k+1)},\ \hat{\mathbf{s}} = \mathbf{s}^{(k+1)},\ \hat{\mathbf{d}} = \mathbf{d}^{(k+1)}$, and $z^* = z(\mathbf{x}^{(k+1)})$.
15: **end if**
16: **end while**
17: **end while**
18: **end while**
19: Set $\overline{z} = z^*$ and $(\mathbf{x}, \hat{\mathbf{s}}, \hat{\mathbf{d}}) = (\mathbf{x}^*, \hat{\mathbf{s}}, \hat{\mathbf{d}})$.
20: **Output:** $\overline{z}^*$ and $(\mathbf{x}, \hat{\mathbf{s}}, \hat{\mathbf{d}})^*$.

5.2. Large sized problems

Second, we compare $\overline{z}$ to $\overline{z}_{\text{JH}}$ and $\overline{z}_{\text{L}}$ on ten large sized problems: problems 1–6 are randomly generated new instances whereas problems 7–10 are benchmark instances [4]. Data for problems 1–6 are given in Table 4. Table 5 reports the same results as Table 3 but for large instances.

Table 5 suggests that the upper minimal total cost bounds to each of problems 8–10 found by all three methods are identical, whereas $\overline{z}$ outperforms z_{JH} for problems 1–7 and z_{L} for problems 1–4. Moreover, for the large size instances, the largest percent increases Δ_{L} and Δ_{JH}, which are 39.90% and 41.70%, are smaller than for the first set of instances, where they reached 94.90% and 90.50%, respectively. This is most likely because the search space is much larger and reaching areas that contain the global maximum bound becomes more challenging. The average CPU time of H, over the ten large size instances, is almost twice as large as the average run time of H_{JH}, further indicating the size of the search space. It is indeed generally difficult for any exact optimal technique to find the exact upper minimal total cost bound for large size instance of the TPVDS even when allocated hours of run time.

TABLE 2. Data for small-sized instances.

#	1	2	3	4	5
$\mathbf{c}$	$\begin{pmatrix} 22 & 10 & 130 \\ 40 & 25 & 39 \end{pmatrix}$	$\begin{pmatrix} 17 & 12 & 100 \\ 25 & 20 & 42 \end{pmatrix}$	$\begin{pmatrix} 15 & 12 & 120 \\ 45 & 30 & 22 \end{pmatrix}$	$\begin{pmatrix} 15 & 25 & 54 & 5 & 25 \\ 31 & 21 & 87 & 29 & 46 \\ 2 & 15 & 10 & 60 & 30 \end{pmatrix}$	$\begin{pmatrix} 9 & 50 & 54 & 10 & 90 & 35 \\ 31 & 17 & 87 & 29 & 46 & 26 \\ 10 & 15 & 11 & 60 & 30 & 45 \end{pmatrix}$
$\underline{\mathbf{S}}$	(60 75)	(60 75)	(60 75)	(60 75 100)	(60 75 100 125)
$\overline{\mathbf{S}}$	(120 150)	(120 150)	(120 150)	(120 150 200)	(120 150 200 250)
$\underline{\mathbf{D}}^T$	(45 30 60)	(45 30 60)	(45 30 60)	(45 30 60 25 75)	(45 30 60 70 80 75)
$\overline{\mathbf{D}}^T$	(90 60 120)	(90 60 120)	(90 60 120)	(90 60 120 50 150)	(90 60 120 140 160 150)

TABLE 3. Comparing upper bounds and run times of H, H_{L} and H_{JH} for small instances.

Problem	H_{L}		H_{JH}		H		$\Delta(\%)$	
	$\overline{z}_{\mathrm{L}}$	RT_{L} (s)	$\overline{z}_{\mathrm{JH}}$	$\mathrm{RT}_{\mathrm{JH}}$ (s)	$\overline{z}$	RT (s)	Δ_H	Δ_{JH}
1	7710	1	7710	1.00	10285	2.20	33.40	33.40
2	7530	1	7530	1.00	8990	1.45	19.40	19.40
3	5250	1	5370	1.20	10230	2.13	94.90	90.50
4	8515	1	8515	1.00	10115	1.65	18.80	18.80
5	13 235	1	13 235	1.00	15 060	1.82	13.80	13.80
6	2130	1	3540	1.21	3690	1.30	73.24	4.24
7	6075	1	7725	1.00	7800	1.20	28.40	0.97
8	7410	1	8430	1.50	8430	1.90	13.80	0.00
9	4350	1	4725	1.00	4800	1.00	10.30	1.59
10	10 890	1	11 565	1.43	11 565	1.96	6.20	0.00
11	6960	1	8730	1.50	8730	1.81	25.40	0.00
12	9000	1	10725	1.63	10725	1.89	19.20	0.00
13	4350	1	7725	1.27	7800	1.90	79.30	0.97
14	10 890	1	12 630	1.49	12 630	1.75	16.00	0.00
15	7290	1	8010	1.18	8460	1.84	16.10	5.62
Average		1		1.23		1.72	31.22	12.62

5.3. A real world case

To evaluate the performance of our proposed algorithm H, we solve a real-world yoghurt distribution problem [7]. The yoghurt factory has two warehouses located at Ibbankatuwa and Dambulla, Sri Lanka. Yoghurt products are distributed island wide to all 25 distribution centres. Table 6 reports the total costs and run times of the solutions of H, H_{L}, and H_{JH} along with the best known bound z_{best} reported in Juman *et al.* [7] and its run time $\mathrm{RT}_{\mathrm{best}}$.

For this real-world problem, the new proposed method H finds the largest near-optimal upper minimal total cost bound in comparison H_{L} and H_{JH}. It further improves z_{best}. This improvement occurs at the small CPU time of 78.6 s.

5.4. Sensitivity analysis

Sensitivity analysis helps us understand the impact of variations δ in the demand and variations ψ in supply range on the upper minimal total cost bound $\overline{z}$. We undertake such analysis on the real-world yoghurt distribution problem, where a range of 0 indicates no variation. Figure 1 illustrates the behavior of the upper bound

Table 4. Data for large-sized instances.

#	1	2
$\mathbf{c}$	$\begin{pmatrix} 19 & 12 & 8 & 45 & 30 & 28 & 29 & 40 & 9 & 7 \\ 23 & 45 & 7 & 8 & 30 & 12 & 93 & 39 & 13 & 15 \\ 9 & 73 & 58 & 7 & 48 & 10 & 60 & 14 & 79 & 52 \\ 24 & 33 & 41 & 11 & 39 & 3 & 11 & 55 & 45 & 7 \\ 5 & 17 & 14 & 71 & 9 & 15 & 38 & 4 & 10 & 9 \end{pmatrix}$	$\begin{pmatrix} 15 & 90 & 88 & 75 & 80 & 8 & 15 & 90 & 88 & 75 \\ 80 & 8 & 15 & 90 & 88 & 75 & 80 & 8 & 15 & 90 \\ 88 & 75 & 80 & 8 & 15 & 90 & 88 & 75 & 80 & 8 \\ 15 & 90 & 88 & 75 & 80 & 8 & 15 & 90 & 88 & 75 \\ 80 & 8 & 15 & 90 & 88 & 75 & 80 & 8 & 15 & 90 \end{pmatrix}$
$\underline{\mathbf{D}}$	$\begin{pmatrix} 45 & 30 & 60 & 25 & 150 & 40 & 65 & 55 & 80 & 70 \end{pmatrix}$	$\begin{pmatrix} 45 & 30 & 10 & 25 & 30 & 10 & 45 & 30 & 10 & 45 \end{pmatrix}$
$\overline{\mathbf{D}}$	$\begin{pmatrix} 90 & 60 & 120 & 50 & 300 & 80 & 130 & 110 & 160 & 140 \end{pmatrix}$	$\begin{pmatrix} 90 & 60 & 20 & 50 & 60 & 20 & 90 & 60 & 20 & 90 \end{pmatrix}$
$\underline{\mathbf{S}}$	$\begin{pmatrix} 60 & 75 & 60 & 75 & 60 \end{pmatrix}$	$\begin{pmatrix} 60 & 75 & 60 & 75 & 60 \end{pmatrix}$
$\overline{\mathbf{S}}$	$\begin{pmatrix} 120 & 150 & 120 & 150 & 120 \end{pmatrix}$	$\begin{pmatrix} 120 & 150 & 120 & 150 & 120 \end{pmatrix}$
#	4	5
$\mathbf{c}$	$\begin{pmatrix} 19 & 12 & 8 & 45 & 30 & 28 & 29 & 40 & 9 & 7 \\ 23 & 45 & 7 & 8 & 30 & 12 & 93 & 39 & 15 & 15 \\ 9 & 73 & 58 & 7 & 48 & 10 & 60 & 14 & 79 & 52 \\ 24 & 33 & 41 & 11 & 39 & 3 & 11 & 55 & 45 & 7 \\ 5 & 17 & 14 & 71 & 9 & 15 & 38 & 4 & 10 & 9 \\ 19 & 12 & 8 & 45 & 30 & 28 & 29 & 40 & 9 & 7 \\ 23 & 45 & 7 & 8 & 30 & 12 & 93 & 39 & 15 & 15 \\ 9 & 73 & 58 & 7 & 48 & 10 & 60 & 14 & 79 & 52 \\ 24 & 33 & 41 & 11 & 39 & 3 & 11 & 55 & 45 & 7 \\ 5 & 17 & 14 & 71 & 9 & 15 & 38 & 4 & 10 & 9 \end{pmatrix}$	$\begin{pmatrix} 19 & 12 & 8 & 45 & 30 & 28 & 29 & 40 & 9 & 7 \\ 23 & 45 & 7 & 8 & 30 & 12 & 93 & 39 & 15 & 15 \\ 9 & 73 & 58 & 7 & 48 & 10 & 60 & 14 & 79 & 52 \\ 24 & 33 & 41 & 11 & 39 & 3 & 11 & 55 & 45 & 7 \\ 5 & 17 & 14 & 71 & 9 & 15 & 38 & 4 & 10 & 9 \\ 19 & 12 & 8 & 45 & 30 & 28 & 29 & 40 & 9 & 7 \\ 23 & 45 & 7 & 8 & 30 & 12 & 93 & 39 & 15 & 15 \\ 9 & 73 & 58 & 7 & 48 & 10 & 60 & 14 & 79 & 52 \\ 24 & 33 & 41 & 11 & 39 & 3 & 11 & 55 & 45 & 7 \\ 5 & 17 & 14 & 71 & 9 & 15 & 38 & 4 & 10 & 9 \\ 19 & 12 & 8 & 45 & 30 & 28 & 29 & 40 & 9 & 7 \\ 23 & 45 & 7 & 8 & 30 & 12 & 93 & 39 & 15 & 15 \\ 9 & 73 & 58 & 7 & 48 & 10 & 60 & 14 & 79 & 52 \\ 24 & 33 & 41 & 11 & 39 & 3 & 11 & 55 & 45 & 7 \\ 5 & 17 & 14 & 71 & 9 & 15 & 38 & 4 & 10 & 9 \\ 19 & 12 & 8 & 45 & 30 & 28 & 29 & 40 & 9 & 7 \\ 23 & 45 & 7 & 8 & 30 & 12 & 93 & 39 & 15 & 15 \\ 9 & 73 & 58 & 7 & 48 & 10 & 60 & 14 & 79 & 52 \\ 24 & 33 & 41 & 11 & 39 & 3 & 11 & 55 & 45 & 7 \\ 5 & 17 & 14 & 71 & 9 & 15 & 38 & 4 & 10 & 9 \end{pmatrix}$
$\underline{\mathbf{D}}$	$\begin{pmatrix} 45 & 30 & 60 & 25 & 150 & 40 & 65 & 55 & 80 & 70 \end{pmatrix}$	$\begin{pmatrix} 45 & 30 & 60 & 25 & 75 & 40 & 65 & 55 & 80 & 70 \\ 45 & 30 & 60 & 25 & 75 & 40 & 65 & 55 & 80 & 70 \end{pmatrix}$
$\overline{\mathbf{D}}$	$\begin{pmatrix} 90 & 60 & 120 & 50 & 300 & 80 & 130 & 110 & 160 & 140 \end{pmatrix}$	$\begin{pmatrix} 90 & 60 & 20 & 50 & 150 & 80 & 130 & 110 & 160 & 140 \\ 90 & 60 & 120 & 50 & 150 & 80 & 130 & 110 & 160 & 140 \end{pmatrix}$
$\underline{\mathbf{S}}$	$\begin{pmatrix} 60 & 75 & 100 & 150 & 160 & 60 & 75 & 100 & 150 & 160 \end{pmatrix}$	$\begin{pmatrix} 60 & 75 & 100 & 150 & 160 & 60 & 75 & 100 & 150 & 160 \\ 60 & 75 & 100 & 150 & 160 & 60 & 75 & 100 & 150 & 160 \end{pmatrix}$
$\overline{\mathbf{S}}$	$\begin{pmatrix} 120 & 150 & 200 & 300 & 320 & 120 & 150 & 200 & 300 & 320 \end{pmatrix}$	$\begin{pmatrix} 120 & 150 & 200 & 300 & 320 & 120 & 150 & 200 & 300 & 320 \\ 120 & 150 & 200 & 300 & 320 & 120 & 150 & 200 & 300 & 320 \end{pmatrix}$
#	5	6
	Data as in Problem 4 of Juman and Hoque [4] except	
	$c_{21} = 13, c_{49} = 15, c_{85} = 19.$	$\mathbf{c} = \begin{pmatrix} 20 & 9 & 29 & 41 & 40 & 43 & 6 & 21 & 40 & 11 \\ 8 & 42 & 9 & 15 & 36 & 32 & 37 & 34 & 8 & 10 \\ 17 & 37 & 33 & 16 & 41 & 7 & 33 & 23 & 26 & 15 \\ 31 & 15 & 36 & 33 & 39 & 5 & 32 & 42 & 10 & 26 \\ 29 & 28 & 25 & 4 & 29 & 27 & 36 & 14 & 34 & 28 \\ 37 & 43 & 29 & 29 & 33 & 24 & 43 & 22 & 50 & 41 \\ 21 & 42 & 18 & 28 & 26 & 47 & 14 & 7 & 27 & 16 \\ 44 & 32 & 19 & 39 & 19 & 41 & 10 & 39 & 48 & 34 \\ 26 & 40 & 14 & 38 & 43 & 18 & 36 & 38 & 43 & 26 \\ 15 & 46 & 50 & 43 & 28 & 18 & 29 & 26 & 24 & 42 \end{pmatrix}$

TABLE 5. Comparing upper bounds and run times of H, H_{L} and H_{JH} for large instances.

Problem	H_{L}		H_{JH}		H		$\Delta(\%)$	
	$\overline{z}_{\mathrm{L}}$	$\mathrm{RT_L}$ (s)	$\overline{z}_{\mathrm{JH}}$	$\mathrm{RT_{JH}}$ (s)	$\overline{z}$	RT (s)	Δ_H	Δ_{JH}
1	20 350	1	15 220	2.00	20 850	4.00	2.46	36.99
2	10 600	1	10 600	2.20	14 030	4.29	32.36	32.36
3	18 840	1	18 600	3.00	26 350	8.00	39.90	41.70
4	9085	1	8210	11.00	9405	21.32	3.50	14.60
5	4350	1	4040	5.27	4350	10.21	0.00	7.67
6	5690	1	5600	5.00	5690	9.98	0.00	1.61
7	4240	1	3840	5.22	4240	11.00	0.00	10.42
8	5320	1	5320	6.56	5320	12.02	0.00	0.00
9	5920	1	5920	5.15	5920	11.34	0.00	0.00
10	7520	1	7520	2.68	7520	5.43	0.00	0.00
Average		1		4.81		9.76	7.82	14.54

TABLE 6. Comparative upper bound solutions of a real-world TPVDS.

$\overline{z}_{\mathrm{L}}$	$\overline{z}_{\mathrm{JH}}$	z_{best}	$\overline{z}$	$\mathrm{RT_L}$ (s)	$\mathrm{RT_{JH}}$ (s)	$\mathrm{RT_{best}}(s)$
933 546	933 546	1 376 777	1 553 931	60.51	60.51	8.84

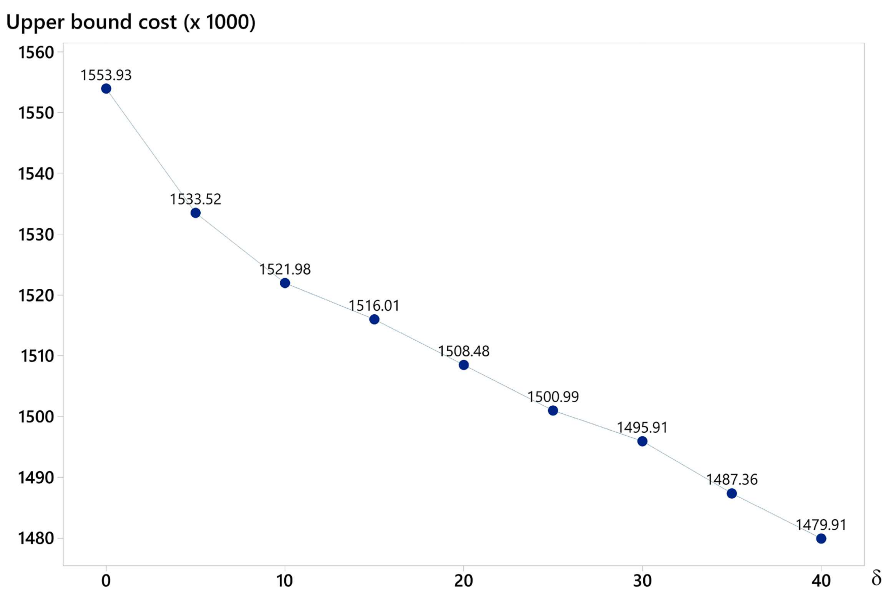

FIGURE 1. Upper bound cost *versus* increment δ of demand range.

cost as the range of demand increases, specifically as δ increases from 0 to 40. The increased range gives more flexibility to the supply end; thus, makes the logistics distribution more efficient, and allowing a better utilization of the supply network. Figure 2 reflect the same behavior when the supply range increases, specifically when ψ increases from 0 to 40. It confirms that the wider the range the more flexibility the supply chain has in channeling the flow; thus, the lower costs.

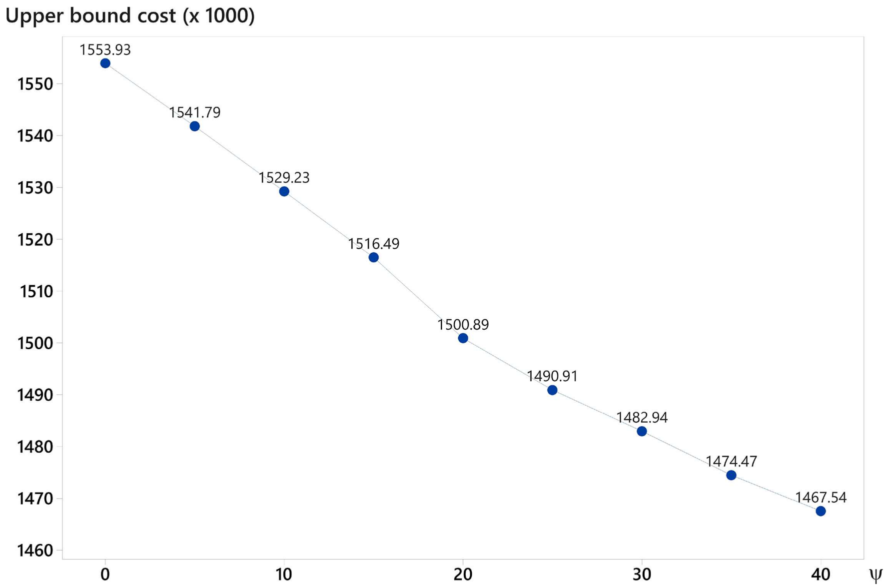

FIGURE 2. Upper bound cost *versus* increment ψ of supply range.

6. Conclusion

In this paper, we investigated the transportation problem with varying demands and supplies (TPVDS). Although, the lower bound of the minimal total costs can be found methodologically, finding the upper minimal total cost bound to the TPVDSII (or TPVDS) within their respective ranges becomes an NP-hard problem. Recent models include transportation and inventory costs at destinations, as they are interrelated factors. For such models, two heuristic techniques calculate the lower and the upper minimal total cost bounds. Herein, we propose a method that matches existing bounds or improves them. Better upper minimal total cost bounds to TPVDS are invaluable to decision makers particularly at a time of fluctuating demand and logistics disruptions. Similarly, the highlighted impact of increments in the demand ranges and supply ranges on the upper minimal total cost bound are of particular importance.

In developing the new method, the demand is assumed to be variable. However, variation in the demand may follow a particular trend. Hence future research might be carried out in extending the extended model, to include an appropriate trend of demand distribution. We intend to devote ourselves in this direction in our future research.

Acknowledgements. The first author acknowledges that this research is supported *via* a research grant from the University of Peradeniya, Sri Lanka.

References

[1] S.K. Das, A. Goswami and S.S. Alam, Multiobjective transportation problem with interval cost, source and destination parameters. *Eur. J. Oper. Res.* **117** (1999) 100–112.

[2] A. Ebrahimnejad, A simplified new approach for solving fuzzy transportation problems with generalized trapezoidal fuzzy numbers. *Appl. Soft Comput.* **19** (2014) 171–176.

[3] F.L. Hitchcock, The distribution of a product from several sources to numerous locations. *J. Math. Phys.* **20** (1941) 224–230.

[4] Z.A.M.S. Juman and M.A. Hoque, A heuristic solution technique to attain the minimal total cost bounds of transporting a homogeneous product with varying demands and supplies. *Eur. J. Oper. Res.* **239** (2014) 146–156.

[5] Z.A.M.S. Juman and M.A. Hoque, An efficient heuristic to obtain a better initial feasible solution to the transportation problem. *Appl. Soft Comput.* **34** (2015) 813–826.

[6] Z.A.M.S. Juman, M.A. Hoque and M.I. Buhari, A study of transportation problem and use of object oriented programming, in 3rd International Conference on Applied Mathematics and Pharmaceutical Sciences (ICAMPS'2013) April 29–30. Singapore (2013).

[7] Z.A.M.S. Juman, M. Masoud, M. Elhenawy, H. Bhuiyan, M.M.R. Komol and O. Battai, A new algorithm for solving uncapacitated transportation problem with interval-defined demands and suppliers capacities. *J. Intell. Fuzzy Syst.* **41** (2021) 625–637.

[8] A. Kaur and A. Kumar, A new approach for solving fuzzy transportation problems using generalized trapezoidal fuzzy numbers. *Appl. Soft Comput.* **21** (2012) 1201–1213.

[9] S.T. Liu, The total cost bounds of the transportation problem with varying demand and supply. *Omega* **31** (2003) 247–251.

[10] P. Liu, L. Yang, L. Wang and S. Li, A solid transportation problem with type-2 fuzzy variables. *Appl. Soft Comput.* **24** (2014) 543–558.

[11] M.R. Safi and A. Razmjoo, Solving fixed charge transportation problem with interval parameters. *Appl. Math. Model.* **37** (2013) 8341–8347.