IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 30, 2019, accepted August 12, 2019, date of publication August 19, 2019, date of current version August 28, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2936095

On the Two-Stage Transportation Problem With Fixed Charge for Opening the Distribution Centers

OVIDIU COSMA[1], DANIELA DĂNCIULESCU[2], AND PETRICĂ C. POP[1]
[1]Department of Mathematics and Computer Science, Technical University of Cluj-Napoca, North University Center of Baia Mare, Baia Mare 430083, Romania
[2]Department Computer Science, University of Craiova, Craiova 200585, Romania

Corresponding author: Petrică C. Pop (petrica.pop@cunbm.utcluj.ro)

ABSTRACT In this paper, we are addressing the two-stage transportation problem with fixed charge for opening the distribution centers, which is an extension of the classical transportation problem. The problem models a distribution network in a two-stage supply chain which involves: manufacturers, distribution centers and customers, and its main characteristic is that a fixed charge for opening the distribution centers is associated, in addition to the variable transportation cost which is proportional to the amount of goods shipped. We describe a novel solution approach for the minimization of total distribution costs: a fast and efficient constructive heuristic algorithm that reduces the solution search space to a subspace with a reasonable size, without losing optimal or sub-optimal solutions by considering a perturbation mechanism that allows us to reconsider discarded feasible solutions that might lead to such solutions. Computational results are reported and discussed for the existing benchmark instances and on a set of instances that contains eight new randomly generated larger instances. The obtained results show that our solution approach is highly competitive as compared to the existing methods from the literature.

INDEX TERMS Transportation system design two-stage fixed-charge transportation problem constructive heuristic algorithms.

I. INTRODUCTION

A. DESCRIPTION OF THE PROBLEM

Supply chains (SCs) are defined as worldwide networks wherein the following actors appear: supplier, manufacturers, distribution centers, retailers and customers. There are several functions the typical SC performs: the procurement of raw materials, the transformation of raw materials into intermediate and end products, as well as the distribution of these products to customers, its main objective being the satisfaction of the customer requirements. Supply Chain Management (SCM) has been widely investigated due to its challenging aspects and its numerous application domains in manufacturing, service industries, transportation, etc., see for more information Masudin [18], Pal and Kant [20], Fu and Zhu [8], etc.

In order to achieve an efficient and effective management of SC systems, increased attention has to be paid to the transportation system design, as it plays an important and central role in this. A typical representation of a SC is as a form of multi-staged structure, while its optimal design has been recognized as NP-hard problem [4].

This paper focuses on a particular supply chain network design problem, namely the two-stage transportation problem with fixed charge for opening the distribution centers, which can be seen as an extension of the classical transportation problem. The problem models a distribution network in a two-stage supply chain which involves: manufacturers, distribution centers and customers and its main characteristic is that a fixed charge is associated for opening the distribution centers, in addition to the variable transportation cost which is proportional to the amount of goods shipped. The objective of the considered transportation problem is to determine the DCs to be opened and to identify and select the routes from manufacturers through the selected distribution centers to the customers satisfying the capacity constraints of the manufacturers and distribution centers in order to meet specific demands of the customers under minimal total distribution costs. In this form, the problem was introduced by Gen *et al.* [9].

The associate editor coordinating the review of this article and approving it for publication was Zhengbing He.

B. LITERATURE REVIEW

Different variants of the two-stage transportation problem have been considered in the literature, depending on the characteristics of the transportation system which models real applications of supply chain network design.

Marin and Pelegrin [16] supposed that the manufacturers and the distribution centers have no capacity restrictions and there exist fixed costs associated to opening the distribution centers and the number of opened distribution centers is fixed and established in advance. In order to solve this version of the two-stage transportation problem, they proposed an algorithm based on a Lagrangean decomposition and branch-and-bound techniques which makes use of the features of the considered transportation problem. Marin [17] studied an uncapacitated version of the problem when both manufacturers and distribution centers acquire fixed costs when they are used, and provided a mixed integer programming model of the problem and lower bounds of the optimal objective values based on different Lagrangian relaxations. Pirkul and Jayaraman [24] considered a multi-commodity, multi-plant, capacitated facility location problem for which they provided a mixed integer programming formulation and an efficient heuristic based on a Lagrangian relaxation of the problem. The same authors in [15] extended their model by also taking into consideration the acquisition of raw material, and presented an efficient heuristic solution approach that utilizes the solution generated from a Lagrangian relaxation of the problem. Amiri [1] proposed a different variant allowing the use of several capacity levels of the manufacturers and distribution centers and described an efficient heuristic approach based on a Lagrangian relaxation of the problem. Calvete *et al.* [2] described a two-levels optimization problem that models the planning of a distribution network that allows one to take into consideration the manner in which decisions made at the distribution stage of the supply chain can affect and be affected by decisions made at the manufacturing stage. They proposed a two-levels mixed integer model of the problem and a metaheuristic solution approach that combines the use of an evolutionary algorithm to control the supply of distribution centers with optimization techniques to determine the delivery from distribution centers to customers and the supply from manufacturers to distribution centers.

Raj and Rajendran [28] considered two scenarios of the problem: the first scenario (Scenario-1) takes into consideration fixed costs associated to the routes in addition to unit transportation costs and unlimited capacities of the distribution centers, while the second one (Scenario 2) which takes into consideration the opening costs of the distribution centers in addition to unit transportation costs. They developed a genetic algorithm (GA) with a specific coding scheme suitable for two-stage problems and as well they presented a set of 20 benchmark instances. Their achieved computational results have been compared to the lower bounds and approximate solutions obtained from a certain relaxation of the problem. Raj and Rajendran in [29] also presented a solution representation that allows a single-stage genetic algorithm (SSGA) to solve it. The main characteristic of these methods is a compact representation of a chromosome based on a permutation. A different genetic algorithm dealing with the two-stage transportation problem with fixed charge associated to the routes from manufacturers to customers was developed by Jawahar and Balaji [14]. Pop *et al.* [27] described, in the case of Scenario-1, a hybrid algorithm that combines a steady-state genetic algorithm with a local search procedure. Recently, Cosma *et al.* [6], [7] developed an efficient multi-start Iterated Local Search (ILS) procedure for the total distribution costs minimization of the TSTP-FC, which comes up with a primary solution, employs a local search procedure with the aim of increasing the exploration, a perturbation mechanism and a neighborhood operator with the aim of diversifying the search and also presented a soft computing approach for solving the two-stage transportation problem with fixed costs associated to the routes that embeds an optimization problem within the framework of a genetic algorithm.

In one of these variants, Molla-Alizadeh-Zavardehi *et al.* [19] considered only one manufacturer. They described an integer programming mathematical formulation of the problem, they proposed a spanning tree-based genetic algorithm with a Prüfer number representation and an artificial immune algorithm for solving it. Some comments concerning the mathematical formulation of the problem were provided by El-Sherbiny [32]. Subsequently, Pintea *et al.* [21], [23] described some hybrid classical approaches and [23] developed an improved hybrid algorithm combining the Nearest Neighbor search heuristic with a local search procedure for solving the two-stage transportation problem with fixed costs. Recently, Pop *et al.* [26] described a novel hybrid heuristic approach obtained by combining a genetic algorithm based on a hash table coding of the individuals with a powerful local search procedure and Cosma *et al.* [5] proposed an efficient hybrid Iterated Local Search (HILS) that constructs an initial solution while using a local search procedure whose aim is to increase the exploration and for the purpose of diversifying the search, a neighborhood structure is used.

Hong *et al.* [13] considered a variant of the fixed-cost transportation problem in a two-stage supply chain network, in which they took into consideration two types of fixed costs: one for opening the distribution centers and the other associated to the routes between manufacturers and distribution centers (DC's) and between DC's and retailers. Some comments concerning the mathematical formulation proposed by Hong *et al.* [13] and a valid formulation of the problem were provided by Sabo *et al.* [30].

There exists yet another version of the two-stage transportation problem with one manufacturer, and it takes into account the environmental impact by reducing the greenhouse gas emissions. This version was introduced by Santibanez-Gonzalez *et al.* [31] in order to deal with a practical application occurring in the public sector. Considering this version of the problem, Pintea *et al.* [22] came up with a

set of classical hybrid heuristic approaches and Pop *et al.* [25] suggested an efficient reverse distribution system for solving the problem.

The variant addressed in this paper considers a two-stage transportation problem with fixed charge for opening the distribution centers, as introduced by Gen *et al.* [9]. The current literature regarding the investigated two-stage transportation problem is scarce. This transportation problem has also been studied by Raj and Rajendran [28], who called it Scenario-2. In both mentioned papers, the authors proposed genetic algorithms based on sequentially getting first a transportation tree for the transportation problem from distribution centers to customers and second a transportation tree for the transportation problem from manufacturers to distribution centers. In both genetic algorithms, the chromosome contains two parts, each encoding one of the transportation trees. Recently, Calvete *et al.* [3] developed a novel hybrid evolutionary algorithm, whose main feature is the use of a new chromosome encoding that provides information about the distribution centers that can be used within the distribution system.

C. OVERVIEW

The aim of this paper is to describe a novel solution approach for solving the two-stage transportation problem with fixed charge for opening the distribution centers. Our constructive heuristic algorithm differs from the existing solution approaches from the literature, it is called Shrinking Domain Search. Its main characteristic is the reduction of the solution search space to a subspace with a reasonable size, without losing optimal or sub-optimal solutions by considering a perturbation mechanism that allows us to reconsider discarded feasible solutions that might lead to such solutions. The results of our computational experiments on the existing benchmark instances from the literature and on a set of instances that contains eight new randomly generated larger instances are presented and analyzed.

Our paper is organized as follows. In Section II, we give some notations and definitions related to the two-stage transportation problem with fixed charge for opening the distribution centers that will be used throughout the paper and present a mathematical model of the problem based on mixed integer linear programming. The novel solution approach for solving the investigated problem is described in Section III. In Section IV we provide implementation details and the computational experiments and the achieved results are presented and discussed in Section V. Finally, we conclude our work and discuss our plans for future work in Section VI.

II. DEFINITION OF THE TWO-STAGE TRANSPORTATION PROBLEM WITH FIXED CHARGE FOR OPENING THE DISTRIBUTION CENTERS

In this section we give a formal definition of the two-stage transportation problem with fixed charge for opening the distribution centers. We start by defining the related sets, decision variables and parameters:

p	the number of manufacturers
q	the number of distribution centers
r	the number of customers
i	manufacturer identifier, $i \in \{1, \ldots, p\}$
j	distribution center identifier, $j \in \{1, \ldots, q\}$
k	customer identifier, $k \in \{1, \ldots, r\}$
w	maximum number of distribution centers that can be opened
D_k	the demand of customer k
S_i	the capacity of manufacturer i
Q_j	the capacity of distribution center $j \in \{1, \ldots, q\}$
F_j	the fixed cost for opening the distribution center j
c^1_{ij}	the unit cost of transportation from manufacturer i to distribution center j
c^2_{jk}	the unit cost of transportation from distribution center j to customer k
x^1_{ij}	the number of units transported from manufacturer i to distribution center j
x^2_{jk}	the number of units transported from distribution center j to customer k
Z_{opt}	the optimal total cost of distribution

Given a set of p manufacturers, a set of q distribution centers (DC's) and a set of r customers with the following properties:

- Each manufacturer $i \in \{1, \ldots, p\}$ has S_i units of supply, each distribution center $j \in \{1, \ldots, q\}$ has a given capacity Q_j and each customer $k \in \{1, \ldots, r\}$ has a demand D_k.
- Each manufacturer may ship to any of the q distribution centers at a transportation cost c^1_{ij} per unit from manufacturer i, where $i \in \{1, \ldots, p\}$, to DC j, where $j \in \{1, \ldots, q\}$;
- Each of the distribution center may ship to any of the r customers at a transportation cost c^2_{jk} per unit from DC j, where $j \in \{1, \ldots, q\}$, to customer k, where $k \in \{1, \ldots, r\}$;
- There exist fixed costs for opening the distribution centers, as well as a limitation on the number of DCs that are allowed to be opened.

The aim of the two-stage capacitated fixed-cost transportation problem with fixed charge for opening the distribution centers is to determine the distribution centers and the routes to be opened and corresponding shipment quantities on these routes, such that the customer demands are fulfilled, all shipment constraints are satisfied, and the total distribution costs are minimized.

An illustration of the investigated two-stage transportation problem with fixed charge for opening the distribution centers is presented in Fig. 1.

By introducing the linear variables: x^1_{ij} representing the amount of units shipped from manufacturer i to DC j, x^2_{jk}

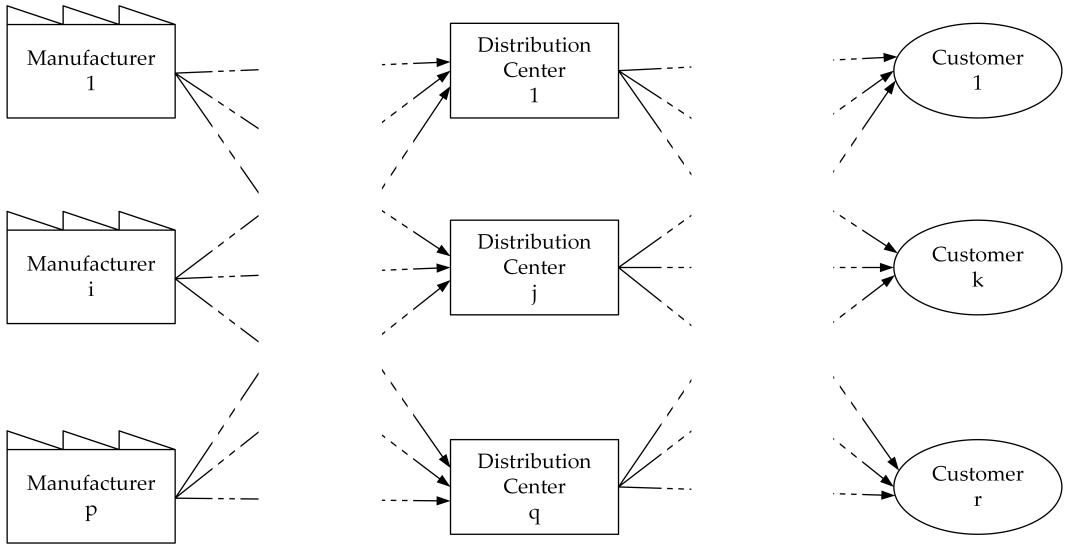

FIGURE 1. Illustration of the two-stage fixed-charge transportation problem with fixed charge for opening the distribution centers.

representing the amount of units shipped from DC j to customer k and the binary variables: z_j is 1 if the distribution center j is opened and 0 otherwise, then the two-stage transportation problem with fixed-charge for opening the distribution centers can be modeled as the following mixed integer linear programming problem described by Calvete *et al.* [3]:

$$\min \sum_{i=1}^{p}\sum_{j=1}^{q} c_{ij}^1 x_{ij}^1 + \sum_{j=1}^{q}\sum_{k=1}^{r} c_{jk}^2 x_{jk}^2 + \sum_{j=1}^{q} F_j z_j$$

$$s.t. \sum_{j=1}^{q} x_{ij}^1 \leq S_i, \quad \forall i \in \{1,\ldots,p\} \tag{1}$$

$$\sum_{j=1}^{q} x_{jk}^2 \geq D_k, \quad \forall k \in \{1,\ldots,r\} \tag{2}$$

$$\sum_{k=1}^{r} x_{jk}^2 \leq Q_j z_j, \quad \forall j \in \{1,\ldots,q\} \tag{3}$$

$$\sum_{j=1}^{q} z_j \leq w \tag{4}$$

$$\sum_{i=1}^{p} x_{ij}^1 = \sum_{k=1}^{r} x_{jk}^2, \quad \forall j \in \{1,\ldots,q\} \tag{5}$$

$$x_{ij}^1 \geq 0, \quad \forall i \in \{1,\ldots,p\},\ \forall j \in \{1,\ldots,q\} \tag{6}$$

$$x_{jk}^2 \geq 0, \quad \forall j \in \{1,\ldots,q\},\ \forall k \in \{1,\ldots,r\} \tag{7}$$

$$z_j \in \{0,1\}, \quad \forall j \in \{1,\ldots,q\} \tag{8}$$

The objective function minimizes the total distribution cost: transportation per-unit costs and the fixed charges for opening the distribution centers. Constraints (1) guarantee that the quantity shipped out from each manufacturer does not exceed the available capacity, constraints (2) guarantee that the total shipment received from DCs by each customer fulfills its demand, constraints (3) guarantee that the quantity shipped out from each distribution center does not exceed the available capacity, constraint (4) limits the number of distribution centers that can be opened and constraints (5) are the flow conservation conditions and they guarantee that the units received by a DC from manufacturers are equal to the units shipped from the distribution centers to the customers. The last three constraints ensure the integrality and non-negativity of the decision variables.

The considered two-stage transportation problem with fixed charge for opening the distribution centers is a *NP*-hard optimization problem because it extends the fixed-charge transportation problem, which has been shown to be *NP*-hard by Guisewite and Pardalos [11]. That is why in order to tackle the two-stage transportation problem with fixed charge for opening the distribution centers, we proposed an efficient constructive heuristic approach which is going to be described in the next section.

III. AN EFFICIENT CONSTRUCTIVE HEURISTIC ALGORITHM FOR SOLVING THE TWO-STAGE FIXED-CHARGE TRANSPORTATION PROBLEM

The difficulty of the two-stage fixed-charge transportation problem lies in the multitude of possible solutions. Analyzing all the feasible solutions of the problem is not possible for practical applications, because it would require an exponential computational time. Alternatively, in this paper we propose an efficient algorithm that reduces the solution search space to a subspace with a reasonable size, without losing optimal or suboptimal solutions. This is done by considering a perturbation mechanism that allows us to reconsider discarded feasible solutions that might lead to optimal or sub-optimal solutions.

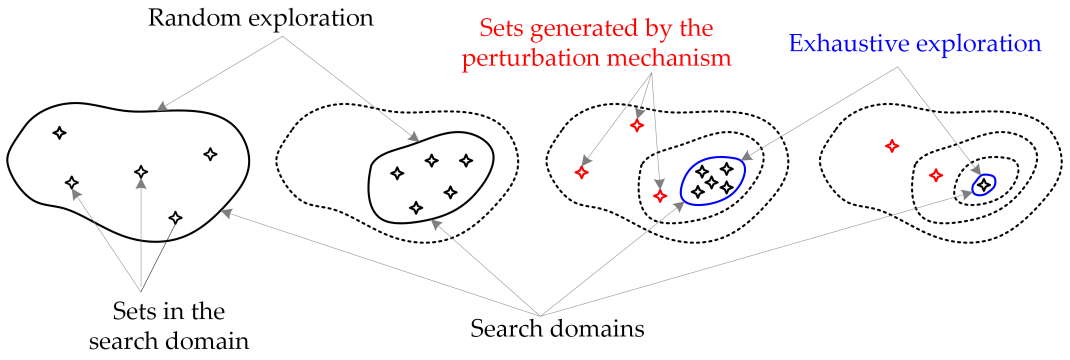

FIGURE 2. The operating principle of the proposed SDS algorithm.

Since the opening of each DC involves a fixed cost, the most important decision the algorithm has to take is to determine the set of DCs that will be used in the distribution solution. Thus, the problem can be naturally decomposed into two subproblems:

- Constructing the optimal distribution solution that uses only a particular set of DCs;
- Finding the set of DCs, based on which the optimal distribution solution can be built. This will be called the *optimal set*.

Our heuristic algorithm is called Shrinking Domain Search (SDS) and it is an iterative algorithm that aims to find the optimal set. Its operating principle is shown in Fig. 2.

The algorithm randomly chooses at each iteration a number of sets of DCs found in a particular search domain, builds distribution solutions based on the chosen sets, and then the search domain is narrowed. Thus, the algorithm ends after a small number of iterations, when a single set of DCs remains in the search domain, with which the best distribution solution can be built. Due to the small number of required iterations, the algorithm can be used successfully for large-scale distribution systems.

For the determination of the search domains, the DCs are classified as ''promising'' or ''wrong'' depending on the cost of the distribution solutions that were previously built. When initializing the algorithm, all DCs are considered promising, but their percentage decreases after each iteration.

The following variables will be used next:

d_{best}	the dimension of the optimal set
q_p	the number of DCs in the promising group
t	the number of basic DC sets produced at an iteration of the algorithm

The search domain corresponding to an iteration consists of all the sets of d_{best} DCs from the promising group. The search domain contains $\binom{q_p}{d_{best}}$ elements. The d_{best} value will be estimated at the initialization step, and the estimate will be updated during the algorithm.

At each iteration of the algorithm (excepting the last ones), the same number of basic distribution solution variants are built (t). Thus, as search domains are reduced, they will be explored more thoroughly, and in the last iterations, when their number of elements drops below t, the search domains will be explored exhaustively. The t constant is an important parameter that influences the efficiency of the algorithm. The lower the value, the higher the algorithm efficiency, but the risk of losing the optimal solution also increases as the search domains will be explored more superficially.

The search domain reduction mechanism does not guarantee that DCs from the optimum set will not be lost. Such DCs may be lost, as they may have been placed only in sets along with disadvantageous DCs, resulting in poor performance distribution solutions. To correct this issue, a perturbation insertion mechanism was created, whereby each DC in the wrong DCs set will be re-analyzed at each iteration by trying to be placed in a new set along with the DCs in the best sets. In order to be able to correct the d_{best} estimate, each iteration will produce auxiliary sets with $d_{best}+1$ and $d_{best}-1$ elements.

Fig. 3 shows the relations between the modules and data structures of the proposed SDS algorithm.

Our proposed algorithm uses the following data structures:

- *Promising DCs* (L1) – a list of the DCs in the promising DCs group. At each iteration of the algorithm, the sets within the search domain will be produced based on this list;
- *Wrong DCs* (L3) – a list of DCs in the wrong DCs group. It is used by the perturbation mechanism, and for adjusting the d_{best} evaluation;
- *Used sets* (L2) - a hash set with all the sets that were produced and eventually evaluated during the execution of the algorithm;
- *Evaluated Sets* (L4) – a list containing the best sets discovered during the algorithm, and the sets evaluated at the last iteration. The first t elements of this list form the *Promising sets* list, which contains the lowest cost sets found from the beginning of the algorithm to the current iteration. The number of elements in this list

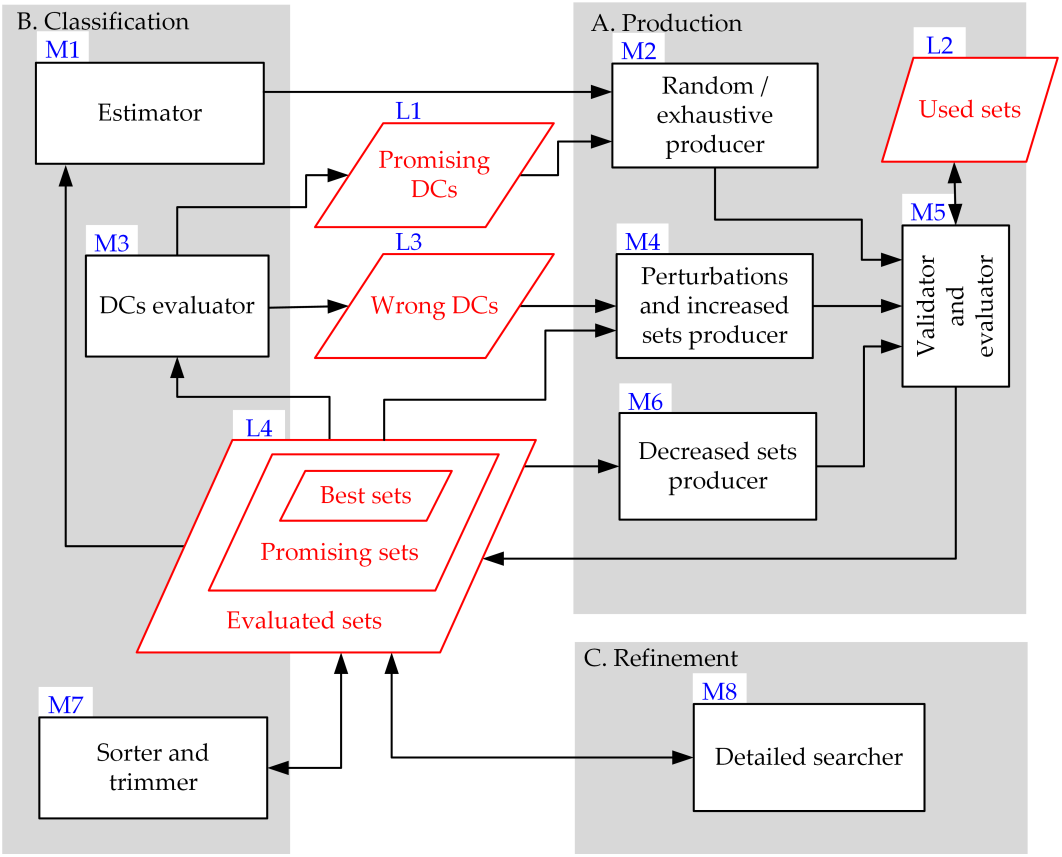

FIGURE 3. The modules and the data structures of the proposed SDS algorithm.

stays constant throughout the algorithm. At the beginning of this list, there are a number of sets composed by *promising DCs* only. Those sets will be called the *Best sets*.

When initializing the algorithm, we evaluate the optimal set dimension (d_{best}), add all available DCs in the *Wrong DCs* list (L3), create the *Promising DCs* list (L1), the *Evaluated sets* list (L4) and the *Used sets* hash set (L2).

At each iteration of the algorithm, the following blocks are run in sequence: *Production* (A) and *Classification* (B). The *Refinement* block (C) is run only once, at the end of the algorithm. The first block generates and evaluates new sets, the second one processes the results, and the third performs a finer search around the best solutions discovered during the algorithm.

The *Production* block (A) contains three types of producers (M2, M4 and M6) for feeding the *Validator and evaluator* module (M5). All the sets generated throughout the algorithm are kept in a hash set (*Used sets* L2). So any duplicate can be efficiently recognized and rejected. This mechanism could be implemented because the algorithm reaches the solution in a very small number of iterations, evaluating a relatively small number of sets. For this reason, it could be tested on large-scale distribution systems.

The *Random / exhaustive producer* module (M2) generates a fixed number of sets, combining the DCs in the *Promising DCs* list (L1). All the sets generated by this module have the same dimension (d_{best}). The d_{best} estimate will be updated after each iteration by the *Estimator* mode (M1). The *Perturbations and increased sets producer* module (M4) creates new sets by combining DCs from the *Wrong DCs* list (L3) with the sets in the *Promising sets* list (L4). The perturbation mechanism is crucial in our algorithm because some DCs may be wrongly classified by the *DCs Evaluator* module (M3) because they were placed in sets only together with disadvantageous DCs. Through this mechanism, all these DCs are given a chance to return to the *Promising DCs* list, at each iteration. The perturbation mechanism tries to produce a new set for every DC from the *Wrong DCs* list by replacing an item in a set from the *Promising sets* list. Each wrong DC will be inserted into a best-ranked set from this list. Equally important is the mechanism for adjusting the d_{best} estimation. For this purpose, a new set will be created for each wrong DC by adding it to a top-ranked set that is retrieved from the *Best sets* list. Thus, sets with an increasing number of DCs can be created. The *Decreased sets producer* module (M6) creates smaller sets by randomly removing one element of the sets from the *Best sets* list.

The *Validator and evaluator* module (M5) attempts to build the best distribution solution that uses only the DCs in each produced set, with a fast heuristic algorithm. Next, we will refer to the cost of that distribution solution, by the cost of

the corresponding set. The evaluated set will be discarded if its cost is greater than the cost of the last set in the *Promising sets* list. The DCs in the considered set that are not used within the built distribution solution will be removed, thus allowing the set to become a duplicate. For this reason, the uniqueness of the sets that undergo changes must be verified after the distribution solution has been constructed, and only if they are unique they will be added to the *Evaluated sets* list (L4).

The *Classification* block (B) is used at the end of each iteration of the algorithm. The *Sorter and trimmer* module (M7) sorts the *Evaluated sets* list by the costs of the sets, and then only the best elements are retained. These elements form the *Promising sets* list, the size of which remains constant. The *DCs evaluator* module updates the contents of the *Promising DCs* and *Wrong DCs* based on the *Promising sets* list. The first DCs encountered when scrolling through the *Promising sets* list will be added to the *Promising DCs* list and the others remain in the *Wrong DCs* list. The percentage of promising DCs decreases at each iteration, which assures the completion of the algorithm.

The *Estimator* module (M1) evaluates the quality of each set dimension by taking the information from the *Best sets* list. The quality of each dimension is estimated according to the number of occurrences in the Best sets list and the positions in which they appear.

The *Refinement* block (C) is required because the *Validator and evaluator* module uses a fast heuristic algorithm to build distribution solutions. Thus, there is no guarantee that the obtained solutions are optimal. The *Detailed searcher* module (M8) performs a fine search around the solutions built for the first sets from the *Best sets* list.

IV. IMPLEMENTATION DETAILS

The following additional notations will be used in the implementation description:

Z_{best}	cost of the best found set;
Z_{worst}	cost of the last set in the *Promising sets* list;
Z_s	cost of set s;
d_s	dimension of set s;
n_{best}	number of elements in the *Best sets* list;
a	percent of promising DCs from the total number of DCs;
b	the rate of decreasing the percent of promising DCs (a).

We start with the description of the initialization step shown in Algorithm 1. Based on preliminary computational experiments, the following parameters were used: the initial number of basic DC sets produced at the first iteration of the algorithm t *initial value* $= 6 \times q$, the initial percent of promising DCs from the total number of DCs a *initial value* $= 0.5$. The call on line 9 produces and evaluates a first collection of sets and the call on line 10 enters the central part of the algorithm. The d_{best} is estimated based on the total demand of the customers and the minimum capacity of the distribution centers as follows:

$$d_{best} = \begin{cases} \dfrac{sumD}{minQ} + 1 & \text{if sumD} \leq \text{minQ} \\ \dfrac{sumD}{minQ} + 2 & \text{if sumD} > \text{minQ} \end{cases}$$

where $sumD = \sum_{k=1}^{r} D_k$ and $minQ = \min_{j \in \{1,\ldots,q\}} Q_j$.

If the estimated value for d_{best} exceeds w, then the initial value of d_{best} will be w.

The *Production* procedure shown in Algorithm 2 produces multiple sets based on the DCs in the *DCsList* parameter. The second parameter indicates whether or not perturbations should be produced and will receive a false value only at the first call, performed in the initialization procedure. The actual production is performed by the *RandomProducer* procedure shown in Algorithm 3, that generates at most t sets. If perturbations are required, then the *IncreasedSetsProducer* procedure shown in Algorithm 4 is called for every DC in the *WrongDCs* list. This procedure generates also increased sets that are useful for the d_{best} estimation.

The *RandomProducer* procedure shown in Algorithm 3 produces sets of d_{best} elements with DCs taken from the *DCsList* parameter. The generated sets are sent to the *ValidateAndEvaluate* procedure shown in Algorithm 6. If there are too few elements in the *DCsList* to produce t sets of d_{best} elements, then all the possible sets will be created systematically in the *for each* loop. Otherwise a number of t random sets will be produced. The procedure uses a working list of shuffled DCs, and avoids adding the same DC in a set more than once.

The *IncreasedSetsProducer* procedure shown in Algorithm 4 is trying to produce a set outside the search domain and a set of $d_{best} + 1$ elements. Both will contain the DC specified by the *wrongDC* parameter. For creating the set outside the search domain, the *Promising sets* list is searched for the best set in which one of the DCs can be replaced with the *wrongDC*. The process stops when a new valid set is created and evaluated, or when the end of the list is reached. For creating the set with increased dimension the *Best sets* list is searched for the best set to which the *wrongDC* can be added to form a new valid set.

This procedure is called in a loop in the *Production* procedure shown in Algorithm 2, for trying to put each wrong DC in the best possible set, by replacing an element and by increasing the set dimension.

The *DecreasedSetsProducer* procedure shown in Algorithm 5 creates all the possible sets by eliminating one DC from the sets in the *Best sets* list. All the new generated sets are validated and evaluated.

The *ValidateAndEvaluate* procedure shown in Algorithm 6 is called after creating each new set. The validation process has two stages. In the first stage, it is checked that the set specified by the parameter s has not been used before. The *Used sets* hash set is used for this purpose. The second stage is controlled by the *verify* parameter. At this stage, it is checked

Algorithm 1 Initialization

```
procedure Initialization
    Z_best ← Z_worst ← ∞
    t ← t initial value
    a ← a initial value
    estimate d_best
    WrongDCs ← new list filled with all the DCs in the distribution system
    EvaluatedSets ← new empty list
    UsedSets ← new empty hash set
    PRODUCTION(WrongDCs, false)
    SDSEARCH
end procedure
```

Algorithm 2 Production

```
procedure PRODUCTION(DCsList : list, perturbations : boolean)
    RANDOMPRODUCER(DCsList)
    if perturbations then
        while WrongDCs size > 0 do
            remove one element r from WrongDCs list
            INCREASEDSETSPRODUCER(r)
        end while
        DECREASEDSETSPRODUCER
    end if
    sort EvaluatedSets list
end procedure
```

Algorithm 3 RandomProducer

```
procedure RANDOMPRODUCER(DCsList : list)
    if C(DCsList size, d_best) ≤ t then
        for each distinct combination c of d_best elements in DCsList do
            s ← new set created with c elements
            VALIDATEANDEVALUATE(s, true)
        end for
    else
        WorkingList ← new empty list
        i ← 0
        while i < t do
            shuffle DCsList
            add DCsList elements to WorkingList
            while WorkingList contains d_best distinct elements do
                remove next d_best unique elements from WorkingList
                create a new set s with the removed elements
                if VALIDATEANDEVALUATE(s, true) then
                    i ← i + 1
                end if
            end while
        end while
    end if
end procedure
```

whether the total capacity of the DCs in set s is large enough to meet all customer demands, according to the next relation:

$$\sum_{j \in s} Q_j \geq \sum_{k=1}^{r} D_k$$

By calling the *BuildDistributionSolution* procedure, a distribution solution is built that uses only DCs in the set s. The DCs that are not used in the distribution solution are removed from the set, which may result in duplicates. For this reason, it is necessary to verify the uniqueness of the set also after

Algorithm 4 IncreasedSetsProducer

```
1: procedure IncreasedSetsProducer(WrongDC : int)
2:     for each set s in PromisingSets list do
3:         if WrongDC ∈ s then
4:             continue
5:         end if
6:         c ← clone of s
7:         for each DC d in c, in random order do
8:             replace d with WrongDC
9:             if ValidateAndEvaluate(c, true) then
10:                goto increasedSets
11:            end if
12:            restore c
13:        end for
14:    end for
15:    increasedSets :
16:    for each set s in BestSets list do
17:        if d_s = w then
18:            continue
19:        end if
20:        c ← clone of s
21:        add WrongDC to c
22:        if ValidateAndEvaluate(c, false) then
23:            return
24:        end if
25:    end for
26: end procedure
```

Algorithm 5 DecreasedSetsProducer

```
1: procedure DecreasedSetsProducer
2:     for each set s in BestSets list do
3:         for each DC d in s do
4:             c ← clone of s
5:             remove d from c
6:             ValidateAndEvaluate(c, true)
7:         end for
8:     end for
9: end procedure
```

the procedure call. The evaluated valid sets are finally added to the *Evaluated sets* list only if they are better than the last promising set: $Z_s < Z_{worst}$.

The *SDSearch* procedure shown in Algorithm 7 is the core of the algorithm. Its main loop reduces the search domain at each iteration, by reducing the percent of promising DCs (a) and as a consequence the number of DCs in the promising group q_p. The rate (b) parameter controls the number of iterations. With a greater rate the algorithm ends faster, but the danger of losing the optimal solution increases, because the search domain narrows too much in a single step. The results published in this paper were obtained with rate $b = 1.1$. The *UpdatePromisingDCs* procedure call updates the two lists (*PromisingDCs* and *WrongDCs*), at each iteration. The *for* loop evaluates the quality of all the dimensions of the sets in the *Best sets* list. The quality estimate takes into consideration the number of occurrences in the *Best sets* list, and the positions of the occurrences. The d_{best} estimation is updated on line 16. The estimate will be used by the producers to generate new sets.

The *UpdatePromisingDCs* procedure shown in Algorithm 8 moves q_p DCs from the *WrongDCs* list to the *PromisingDCs* list. The DCs are taken from the first sets in the sorted *PromisingSets* list. The n_{best} variable is also updated.

The *BuildDistributionSolution* procedure builds a distribution solution in r steps. Every step is looking for a better supply for one of the customers, in the conditions created by the decisions taken in the previous steps, when some of

Algorithm 6 ValidateAndEvaluate

```
function ValidateAndEvaluate(s : set, verify : boolean) : boolean
    if s ∈ usedSets then
        return false
    end if
    add s to usedSets
    if verify and s is invalid then
        return false
    end if
    BuildDistributionSolution(s)
    if (s was modified and s ∈ UsedSets) or Z_s > Z_worst then
        return false
    end if
    add s to the EvaluatedSets list
    return true
end function
```

Algorithm 7 SDSearch

```
procedure SDSearch
    while (q_p ← q × a) > d_best do
        a ← a/b
        UpdatePromisingDCs
        SetsDims ← new list of structures {dimension, quality}
        for i ← 0 to n_best do
            s ← element i from BestSets
            score ← n_best − i
            find element e in SetsDims for which d_e = d_s
            if not found then
                add a new element {d_s, score} to SetsDims
            else
                e.quality ← e.quality + score
            end if
        end for
        d_best ← dimension of the highest quality element from SetsDims
        Production(PromisingDCs, true)
        trim EvaluatedSets list to the first t elements
        fill WromgDCs list with all the DCs
    end while
end procedure
```

the capacity of the manufacturers and DCs was consumed. The demand of each customer is solved in one or more steps. At each step, the most advantageous supply route is sought, depending on the unit costs of the transport routes, and the remaining capacities of the manufacturers and DCs. If the route found cannot ensure the customer's entire demand due to limited capacities at manufacturers and DCs, then a new search step for the remaining quantity follows. If a particular customer can not supply all the required quantity on the cheapest possible route, then a *NotBest* flag is set for that solution. This flag indicates that the solution might be improved. At the end of the procedure, the unused DCs from the evaluated set are removed and the fixed costs of the remaining DCs are added to the total cost of the distribution solution. This fast constructive procedure would find the optimal solution if the capacities of DCs and manufacturers were not limited.

The Detailed search module contains a *DetailedSearch* procedure that is required because the *BuildDistribution Solution* does not guarantee the best solution due to the limited capacities of the manufacturers and distribution centers. Before applying this procedure, the *Promising sets* list is reduced to the first t_{fin} (t final) elements. The *DetailedSearch* procedure contains a loop that ends after N_{it} consecutive iterations that do not improve the best known solution. Every iteration of this loop changes the order in which customers will

Algorithm 8 UpdatePromisingDCs

```
1:  procedure UpdatePromisingDCs
2:      PromisingDCs ← new empty list
3:      n_best ← 0
4:      for each set s in PromisingSets list do
5:          for each DC d in s do
6:              if d ∉ PromisingDCs list then
7:                  move d from WrongDCs list to PromisingDCs list
8:                  if PromisingDCs list size = q_p then
9:                      return
10:                 end if
11:             end if
12:         end for
13:         n_best ← n_best + 1
14:     end for
15: end procedure
```

be served. There follows an inner loop that repeats for each *Promising Sets* item that has the *NotBest* flag set. This loop builds an initial solution using the *BuildDistributionSolution* procedure. Next an iterative process attempts to replace some of the solution routes with better variants. This process ends when the last iteration no longer improves the solution. At each iteration two lists are created: *MList* and *DList*. In *MList* all the routes between manufacturers and distribution centers for which $x_{ij}^1 > 0$ are added, and in *DList* all the routes between distribution centers and customers for which $x_{jk}^2 > 0$ are added. Next, all the sets consisting of two elements taken from the DList ($l1$ and $l2$) and an element taken from the *MList* ($l3$) are tested. All pairs of manufacturer-distribution center-customer routes of which $l1$ and $l2$ are part are generated, then they are sent one at a time along with $l3$ to a procedure named *ModifyPlan*, to look for a better alternative.

The *ModifyPlan* procedure searches a replacement for the routes it gets through the parameters. The three routes are canceled, after which a more advantageous supply option is sought, under the new conditions created. The first two routes specify complete manufacturer-distribution center-customer routes, and the third only specifies a link between a manufacturer and a distribution center. For replacing the two complete routes, a method similar to the one described in the *BuildDistributionSolution* procedure is used. To replace the incomplete route, the distribution center's links with all manufacturers are evaluated and the most advantageous one is chosen. The need can be ensured in several steps, from different manufacturers. If a better solution is not attained, then the initial solution is restored.

In Fig. 4 and 5 we present the manner in which a solution can be modified by the *DetailedSearch* procedure. Fig. 4a shows an initial solution and three links that can be chosen ($l1, l2$ and $l3$). Fig. 4b shows three routes ($r1, r2$ and $r3$) that can be created and transmitted to the *ModifyPlan* procedure. Fig. 5a shows the new routes that can be discovered

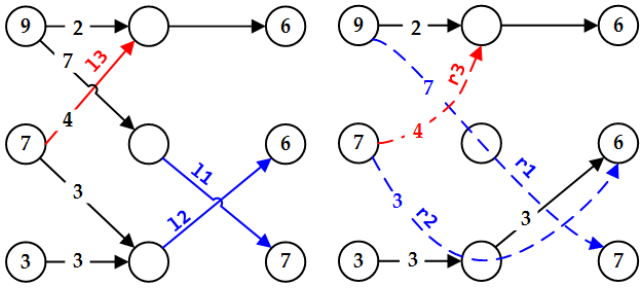

FIGURE 4. Illustration of an initial solution (a) and three routes that can be created and transmitted to the *ModifyPlan* procedure (b).

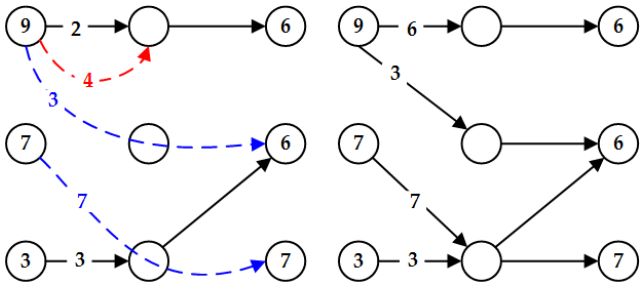

FIGURE 5. Illustration of the new routes (a) and the new solution of the problem (b).

by this procedure, and Fig. 5b presents the new solution. Through such transformations, the solution provided by the *BuildSolution* procedure can be improved which can lead even to the optimal solution if a large enough number of iterations are performed.

The performance of our proposed SDS algorithm depends on the following parameters:

- t which determines how detailed each search domain is explored. The lower the value, the higher the efficiency of the algorithm, but it increases the risk of missing the optimum solution;

- a initial value, which determines how steep the first narrowing of the search domain is. The smaller the value, the faster the algorithm reaches the solution, and the fewer search domains are explored;
- b which determines the speed of convergence (the speed at which the search domain narrows from one iteration to another). Higher rates can cause important DCs to be lost, and low ones lead to an unwarranted increase in execution time. By adjusting this parameter, there is actually a compromise between safety and efficiency.

V. COMPUTATIONAL RESULTS

In this section we present our achieved computational results in order to assess the effectiveness of our proposed solution approach for solving the two-stage transportation problem with fixed charge for opening the distribution centers.

We conducted our computational experiments for solving the two-stage transportation problem with fixed charge for opening the distribution centers on two sets of benchmark instances used in the literature and on a set of instances that contains eight new randomly generated larger instances:

- The first set of instances contains 7 instances and it was used by Gen *et al.* [9] and Raj and Rajendran [28]. In these instances, the number of manufacturers ranges from 3 to 40, the number of DCs ranges from 4 to 70, the number of customers ranges from 5 to 100 and are characterized by the fact that the total supply of the manufacturers, the total capacity of the distribution centers and the total demand of the customers are equal. The instances are available at the address https://sites.google.com/site/fctpdataset/.
- The second set of instances contains 16 instances and it has been randomly generated by Calvete *et al.* [3] and in which the total supply of manufacturers, the total capacity of the DCs and the total demand of the customers are not equal. The manufacturers, distribution centers and customers were randomly generated in the square $[-400, 400] \times [-400, 400]$ according to a discrete uniform distribution. The transportation costs are given by the euclidean distance and the fixed costs for opening a distribution center are given by the following formula: $F_j = 50 \times Q_j$, where $j \in \{1, \ldots, q\}$. Customer demands D_k are randomly chosen as integers in the interval [10, 30]. Concerning the capacities of the manufacturers and distribution centers, there are two different classes of instances. In the first class of instances, the capacities of the manufacturers and distribution centers are given by: $30 \times r$, meaning that each manufacturer and distribution center can supply all the customers. In the second class of instances, there is a reduced number of large manufacturers and distribution centers that may supply all given customers and the remaining ones have smaller capacities. In this case 20% of the manufacturers and distribution centers have a capacity equal to $30 \times r$ and the remaining ones have a capacity equal to $6 \times r$.
- The third set of problems contains 8 new randomly generated instances of larger sizes. The files containing the instances are available at the address: https://sites.google.com/view/tstp-instances/ and have been generated in the same way Calvete *et al.* [3] did. The characteristics of these new instances are displayed in Table 4.

Our solution approach for solving the two-stage transportation problem with fixed charge for opening the distribution centers has been implemented in Java 8 and we performed 5 independent runs for each instance as it was done by Calvete *et al.* [3], on a PC with Intel Core i5-4590 3.3GHz, 4GB RAM, Windows 10 Education 64 bit operating system.

Based on preliminary computational experiments the parameters involved within our *Shrinking Domain Search* algorithm have been set as follows: the number of basic DC sets produced at each iteration of the algorithm $t = 6 \times q$, the initial percent of promising DCs from the total number of DCs $a = 0.5$, the rate of decreasing the promising DCs procent $b = 1.1$, the final number parameter $t_{fin} = 30$ and the maximum number of iterations of the *DetailedSearch* procedure $N_{it} = 50$.

In order to study the performance of our proposed solution approach, we compared it with the existing results from the literature for the considered test instances: the hybrid evolutionary algorithm developed by Calvete *et al.* [3] and the genetic algorithms described by Gen *et al.* [9] and Raj and Rajendran [28]. The obtained computational results are presented in Tables 1-4.

The first results that we report have been obtained in the case of seven instances, denoted by $P_1, \ldots, P_7$, introduced by Gen *et al.* [9] and used by Raj and Rajendran [28]. The first column of the table displays the name of the instance, the next three columns contain the characteristics of the problem: the number of manufacturers (p), the number of distribution centers (q) and the number of customers (r). Next column provides the optimal solution of the problem and the last four columns provide the best solutions achieved by the genetic algorithms developed by Gen *et al.* [9] and Raj and Rajendran [28], the hybrid evolutionary algorithm described by Calvete *et al.* [3] and by our novel solution approach.

Analyzing the displayed results from Table 1, we can observe that our proposed solution approach has a better computational performance than the genetic algorithms considered by Gen *et al.* [9] and by Raj and Rajendran [28] and provides the same solution in all five runs of the computational experiments for all seven instances, as the one provided by Calvete *et al.* [3], and which coincides with the optimal solution.

Tables 2 and 3 provide the results of the computational experiments in the case of the two classes of instances introduced by Calvete *et al.* [3]. Both tables have the same structure: the first column of the tables give the name of the test instance, the next two columns provide the optimal solution of the problem Z_{opt} provided by the professional optimization

TABLE 1. Performance evaluation of our solution approach and the existing methods on the set of instances introduced by Gen *et al.* [9].

Instance	p	q	r	Z_{opt}	Z_{best} [9] Gen et al.	Z_{best} [28]	Z_{best} [3]	Our solution approach
P_1	3	4	5	1089	1089	1089	1089	1089
P_2	4	5	10	2283	2283	2283	2283	2283
P_3	4	5	15	2527	2527	2527	2527	2527
P_4	8	10	20	2886	2886	2886	2886	2886
P_5	10	20	40	2924	2952	2924	2924	2924
P_6	15	25	50	2877	2962	2877	2877	2877
P_7	40	70	100	2925	3136	2933	2925	2925

TABLE 2. Computational results for the first class of instances introduced by Calvete *et al.* [3].

Instance	LINGO		HEA Calvete et al. [3]			Our solution approach				
	Z_{opt}	T_{opt}	Z_{best}	T_{avg}	It_{avg}	Z_{best}	T_{best}	T_{avg}	It_{best}	It_{avg}
$P_{1,1}$	295703	1	295703	0.05	1.0	295703	< 0.001	< 0.001	1	1.0
$P_{2,1}$	500583	1	500583	0.08	1.6	500583	< 0.001	< 0.001	1	1.0
$P_{3,1}$	803019	7	803019	0.81	4.2	803019	0.016	0.028	1	1.2
$P_{4,1}$	1230358	144	1230358	1.78	9.0	1230358	0.031	0.040	1	1.2
$P_{5,1}$	1571893	2633	1571893	6.88	11.4	1571893	0.218	0.269	1	1.8
$P_{6,1}$	2521232	> 2 hours	2521232	11.73	13.4	2521232	0.312	0.438	1	2.6
$P_{7,1}$	3253335	> 2 hours	3253335	56.73	23.4	3253335	1.860	2.457	1	1.6
$P_{8,1}$	4595835	> 2 hours	4595835	38.39	9.6	4595835	3.188	3.347	1	1.0

TABLE 3. Computational results for the second class of instances introduced by Calvete *et al.* [3].

Instance	LINGO		HEA Calvete et al. [3]			Our solution approach				
	Z_{opt}	T_{opt}	Z_{best}	T_{avg}	It_{avg}	Z_{best}	T_{best}	T_{avg}	It_{best}	It_{avg}
$P_{1,2}$	228306	1	228306	0.20	5.6	228306	1.25	18.56	11	11.0
$P_{2,2}$	348837	0	348837	0.23	4.8	348837	0.04	0.10	11	11.0
$P_{3,2}$	507934	2	507934	2.55	12.0	507934	0.84	1.54	18	18.0
$P_{4,2}$	713610	4	713610	4.17	12.6	713610	0.11	0.13	2	2.8
$P_{5,2}$	985628	33	985628	20.46	39.4	985628	1.75	2.27	8	10.6
$P_{6,2}$	1509476	51	1509476	43.70	62.2	1509476	2.56	3.27	8	10.4
$P_{7,2}$	1888252	305	1888252	120.92	54.0	1888252	16.81	24.72	9	13.6
$P_{8,2}$	2669231	> 2 hours	2669231	154.42	35.2	2669231	10.59	30.07	3	10.2

software LINGO together with the corresponding execution time, next three columns display the best solution obtained in all five runs of the computational experiments performed by Calvete *et al.* [3] together with the corresponding average computational time required to reach the best solution and the average in the five runs of the iteration at which the best solution appeared. Finally the last five columns contain information concerning our novel solution approach: the best solution obtained in all five runs of the computational experiments, the corresponding best computational time for achieving the solution, the average computational time for achieving the best solution, the iteration at which the best solution appears and the average in the five runs of the iteration at which the best solution appears. The computational times are reported in seconds with the exception of problems $P_{6,1}$, $P_{7,1}$, $P_{8,1}$ and $P_{8,2}$ when the execution time for LINGO is more than two hours.

Analyzing the displayed results from Tables 2 and 3, we can observe that our proposed solution approach provides the same solution in all five runs of the computational experiments, as the one provided by Calvete *et al.* [3], and which coincides with the optimal solution obtained using LINGO. Regarding the computational times, our algorithm is faster in comparison to the hybrid evolutionary algorithm described by Calvete *et al.* [3] and the explanation is based on the manner in which the combinations of distribution centers that are going to be opened are searched: while Calvete *et al.* [3] is using a genetic algorithm that uses simplex for fitness evaluation for this operation, we are using an efficient and fast heuristic procedure.

In Table 4, we present the characteristics of the new randomly generated instances of larger sizes and the computational results achieved by our proposed SDS algorithm. The first column contains the name of the instances, the next four columns display the characteristics of the instances: the number of manufacturers, distribution centers, customers and the maximum number of distribution centers that can be opened, and the last six columns contain the following results achieved by our SDS heuristic algorithm: the best solution, the average solution, the corresponding best computational time for achieving the solution, the average computational times for achieving the solution in all the five runs of the computational experiments, the iteration at which the best solution appears and the average in the five runs of the iteration at which the best solution appears. The computational times are reported in seconds.

We can observe that in all the new generated test instances our algorithm provides the same solution in all the five runs

TABLE 4. Computational results for the new test instances.

Instance	Characteristics of the instances				Our solution approach					
	p	q	r	w	Z_{best}	Z_{avg}	T_{best}	T_{avg}	It_{best}	It_{avg}
I_1	150	300	800	50	3600012	3600012	206.79	216.52	18	19.0
I_2	150	300	1000	50	4531394	4531394	260.37	311.11	14	16.8
I_3	200	400	1500	60	6594333	6594333	468.97	721.36	8	12.8
I_4	200	400	2000	60	8828329	8828329	461.28	1020.21	7	16.2
I_5	250	500	2500	70	11055247	11055247	2709.26	2733.78	21	21.5
I_6	250	500	3000	70	13150463	13150463	1423.67	2275.37	8	15.0
I_7	300	600	3500	80	15190167	15190167	1531.77	4439.75	5	15.8
I_8	300	600	4000	80	17266134	17266134	3046.88	5368.18	11	19.8

of the computational experiments, confirming the quality and robustness of our proposed solution approach. These solutions have been achieved within reasonable computational times and within a small number of iterations.

VI. CONCLUSION

This paper proposes an efficient and fast constructive heuristic algorithm for solving the two-stage fixed-charge transportation problem which models an important transportation systems design from manufacturers to customers through distribution centers. Our Shrinking Domain Search algorithm is based on the reduction of the solution search space to a subspace with reasonable size, without losing optimal or sub-optimal solutions, by considering a perturbation mechanism that allows us to reconsider discarded feasible solutions that might lead to such solutions.

Some important features of our proposed method are:

- it is based on the reduction of the solution search space to a subspace with a reasonable size, without losing optimal or sub-optimal solutions by considering a perturbation mechanism that allows us to reconsider discarded feasible solutions;
- it is highly efficient providing the best existing solutions for all the test instances and in all five runs of the computational experiments within short computational times and number of iterations;
- it can be easily adapted to different distribution systems, such as the two-stage transportation problem with fixed costs associated to the routes, etc., confirming its flexibility.

The computational results on the existing benchmark instances from the literature and on a set of instances that contains eight new randomly generated larger instances show that our proposed novel solution approach is highly competitive as compared to the existing methods and outperforms the other approaches for solving the two-stage transportation problem with fixed charge for opening the distribution centers in terms of necessary execution times to find the solutions.

In the future, we plan to use our code as the basis for a parallel implementation and to test our heuristic algorithm on even larger instances.

REFERENCES

[1] A. Amiri, "Designing a distribution network in a supply chain system: Formulation and efficient solution procedure," *Eur. J. Oper. Res.*, vol. 171, no. 2, pp. 567–576, Jun. 2006.

[2] H. I. Calvete, C. Galé, and J. Iranzo, "Planning of a decentralized distribution network using bilevel optimization," *Omega*, vol. 49, pp. 30–41, Dec. 2014.

[3] H. I. Calvete, C. Galé, and J. A. Iranzo, "An improved evolutionary algorithm for the two-stage transportation problem with fixed charge at depots," *OR Spectr.*, vol. 38, no. 1, pp. 189–206, Jan. 2016.

[4] S. Chen, Y. Zheng, C. Cattani, and W. Wang, "Modeling of biological intelligence for SCM system optimization," *Comput. Math. Methods Med.*, vol. 2012, Oct. 2012, Art. no. 769702.

[5] O. Cosma, P. Pop, O. Matei, and I. Zelina, "A hybrid iterated local search for solving a particular two-stage fixed-charge transportation problem," in *Hybrid Artificial Intelligent Systems* (Lecture Notes in Computer Science), vol. 10870. Springer, 2018, pp. 684–693.

[6] O. Cosma, P. Pop, and I. Zelina, "An efficient soft computing approach for solving the two-stage transportation problem with fixed costs," in *Proc. Adv. Intell. Syst. Comput. (SOCO)*, vol. 950, 2019, pp. 523–532.

[7] O. Cosma, P. Pop, and C. P. Sitar, "An efficient iterated local search heuristic algorithm for the two-stage fixed-charge transportation problem," *Carpathian J. Math.*, vol. 35, no. 2, pp. 153–164, 2019.

[8] Y. Fu and J. Zhu, "Big production enterprise supply chain endogenous risk management based on blockchain," *IEEE Access*, vol. 7, pp. 15310–15319, 2019.

[9] M. Gen, F. Altiparmak, and L. Lin, "A genetic algorithm for two-stage transportation problem using priority-based encoding," *OR Spectr.*, vol. 28, no. 3, pp. 337–354, Jul. 2006.

[10] A. M. Geoffrion and G. W. Graves, "Multicommodity distribution system design by benders decomposition," *Manage. Sci.*, vol. 20, no. 5, pp. 822–844, Jan. 1974.

[11] G. Guisewite and P. M. Pardalos, "Minimum concave-cost network flow problems: Applications, complexity, and algorithms," *Ann. Oper. Res.*, vol. 25, no. 1, pp. 75–99, Dec. 1990.

[12] W. M. Hirsch and G. B. Dantzig, "The fixed charge problem," *Naval Res. Logistics Quart.*, vol. 15, no. 3, pp. 413–424, Sep. 1968.

[13] J. Hong, A. Diabat, V. V. Panicker, and S. Rajagopalan, "A two-stage supply chain problem with fixed costs: An ant colony optimization approach," *Int. J. Prod. Econ.*, vol. 204, pp. 214–226, Oct. 2018.

[14] N. Jawahar and A. N. Balaji, "A genetic algorithm for the two-stage supply chain distribution problem associated with a fixed charge," *Eur. J. Oper. Res.*, vol. 194, no. 2, pp. 496–537, Apr. 2009.

[15] V. Jayaraman and H. Pirkul, "Planning and coordination of production and distribution facilities for multiple commodities," *Eur. J. Oper. Res.*, vol. 133, no. 2, pp. 394–408, Jan. 2001.

[16] A. Marín and B. Pelegrín, "A branch-and-bound algorithm for the transportation problem with location of *p* transshipment points," *Comput. Oper. Res.*, vol. 24, no. 7, pp. 659–678, Jul. 1997.

[17] A. Marín, "Lower bounds for the two-stage uncapacitated facility location problem," *Eur. J. Oper. Res.*, vol. 179, no. 3, pp. 1126–1142, Jun. 2007.

[18] I. Masudin, "Location-allocation problems in the perspective of supply chain: Approaches and applications," *Jurnal Teknik Industri*, vol. 20, no. 1, pp. 1–11, Feb. 2019.

[19] S. Molla-Alízadeh-Zavardehi, M. Hajiaghaei-Kesteli, and R. Tavakkoli-Moghaddam, "Solving a capacitated fixed-charge transportation problem by artificial immune and genetic algorithms with a Prüfer number representation," *Expert Syst. Appl.*, vol. 38, no. 8, pp. 10462–10474, Aug. 2011.

[20] A. Pal and K. Kant, "Internet of perishable logistics: Building smart fresh food supply chain networks," *IEEE Access*, vol. 7, pp. 17675–17695, 2019.

[21] C.-M. Pintea, C. P. Sitar, M. Hajdu-Macelaru, and P. Petrica, "A hybrid classical approach to a fixed-charged transportation problem," in *Hybrid Artificial Intelligent Systems* (Lecture Notes in Computer Science), vol. 7208, E. Corchado, Ed. 2012, pp. 557–566.
[22] C.-M. Pintea, P. C. Pop, and M. Hajdu-Macelaru, "Classical hybrid approaches on a transportation problem with gas emissions constraints," in *Soft Computing Models in Industrial and Environmental Applications*, vol. 188. Springer, 2013, pp. 449–458.
[23] C. M. Pintea and P. C. Pop, "An improved hybrid algorithm for capacitated fixed-charge transportation problem," *Logic J. IJPL*, vol. 23, no. 3, pp. 369–378, Jun. 2015.
[24] H. Pirkul and V. Jayaraman, "A multi-commodity, multi-plant, capacitated facility location problem: Formulation and efficient heuristic solution," *Comput. Operations Res.*, vol. 25, no. 10, pp. 869–878, Oct. 1998.
[25] P. C. Pop, C.-M. Pintea, C. P. Sitar, and M. Hajdu-Măcelaru, "An efficient reverse distribution system for solving sustainable supply chain network design problem," *J. Appl. Logic.*, vol. 13, no. 2, pp. 105–113, Jun. 2015.
[26] P. C. Pop, O. Matei, C. P. Sitar, and I. Zelina, "A hybrid based genetic algorithm for solving a capacitated fixed-charge transportation problem," *Carpathian J. Math.*, vol. 32, no. 2, pp. 225–232, Jan. 2016.
[27] P. C. Pop, C. Sabo, B. Biesinger, B. Hu, and G. Raidl, "Solving the two-stage fixed-charge transportation problem with a hybrid genetic algorithm," *Carpathian J. Math.*, vol. 33, no. 3, pp. 365–371, 2017.
[28] K. A. A. D. Raj and C. Rajendran, "A genetic algorithm for solving the fixed-charge transportation model: Two-stage problem," *Comput. Oper. Res.*, vol. 39, no. 9, pp. 2016–2032, Sep. 2012.
[29] K. A. A. D. Raj and C. Rajendran, "A hybrid genetic algorithm for solving single-stage fixed-charge transportation problems," *Technol. Operation Manage.*, vol. 2, no. 1, pp. 1–15, Jan. 2011.
[30] C. Sabo, A. Horvat-Marc, and P. C. Pop, "Commnents on 'A two-stage supply chain problem with fixed costs: An ant colony optimization approach," *Creative Math. Inform.*, vol. 28, no. 2, pp. 183–189, 2019.
[31] E. D. R. Santibanez-Gonzalez, M. G. Robson, and H. P. Luna, "Solving a public sector sustainable supply chain problem: A genetic algorithm approach," in *Proc. Int. Conf. Artif. Intell. (ICAI)*, Las Vegas, NV, USA, 2011, pp. 507–512.
[32] M. M. El-Sherbiny, "Comments on 'solving a capacitated fixed-charge transportation problem by artificial immune and genetic algorithms with a Prüfer number representation' by Molla-Alizadeh-Zavardehi, S. et al. expert systems with applications (2011)," *Expert Syst. Appl.*, vol. 39, no. 12, pp. 11321–11322, Sep. 2012.

OVIDIU COSMA received the B.S. degree in automatic control and computer science and the Ph.D. degree in automatic control from Politehnica University, Bucharest, Romania, in 1986 and 2004, respectively. From 1986 to 1992, he was an Analyst Programmer with the Electronic Computing Center, Baia Mare. In 1992, he began his activity as Assistant Professor with North University, Baia Mare. He is currently an Associate Professor with the Technical University of Cluj-Napoca, the North University Center, Baia Mare. His research interests include optimization algorithms, image processing, and programming languages. He has authored six books, more than 50 research articles, and two inventions.

DANIELA DĂNCIULESCU received the B.S. degree in informatics from the University of Craiova, in 1994, the B.S. degree in accounting and management informatics, in 2009, the M.S. degree in management, in 2011, the Ph.D. degree in cybernetics and economic statistics from the Faculty of Economic Sciences, University of Craiova, in 2003, under the supervision of Prof. N. Vasilescu, the second Ph.D. degree in computer science from the West University of Timisoara, in 2015, under the supervision of Prof. V. Negru, and the Habilitation degree in economic informatics from the A. I. Cuza University of Iaşi, in 2018. She is currently an Associate Professor with the University of Craiova. She has authored a monograph *Information Systems: Modern Applied and Implemented Methods in Java* (Universitaria Publishing House Craiova, 2006). She has authored/coauthored eight course books for students and over 80 papers in peer-reviewed journals, out of which 18 ISI papers and papers in ISI or conference proceedings indexed in well-known international databases.

PETRICĂ C. POP received the B.S. degree in mathematics from the Babes-Bolyai University of Cluj-Napoca, Romania, the M.S. and Ph.D. degrees in operations research from the University of Twente, The Netherlands, and the Habilitation degree in informatics from the Babes-Bolyai University of Cluj-Napoca. He currently serves as a Professor with the Department of Mathematics and Computer Science, Technical University of Cluj-Napoca, the North University Center, Baia Mare. He wrote more than 130 papers from which more than 100 appeared in ISI journals, ISI proceedings, and international journals. His research interests include combinatorial optimization, mathematical modeling, artificial intelligence, and operations research. Several research stays have taken him to Italy, U.K., Japan, France, Austria, Greece, The Netherlands, Spain, and Canada.