INTERNATIONAL JOURNAL OF SYSTEMS SCIENCE: OPERATIONS & LOGISTICS
2018, VOL. 5, NO. 4, 356–373
https://doi.org/10.1080/23302674.2017.1316877

Taylor & Francis
Taylor & Francis Group

Multi-stage supply chain network solution methods: hybrid metaheuristics and performance measurement

Madjid Tavana[a,b], Francisco J. Santos-Arteaga[c,d], Ali Mahmoodirad[e], Sadegh Niroomand[f,†] and Masoud Sanei[g]

[a]Business Systems and Analytics Department, La Salle University, Philadelphia, PA, USA; [b]Business Information Systems Department, Faculty of Business Administration and Economics, University of Paderborn, Paderborn, Germany; [c]School of Economics and Management, Free University of Bolzano, Bolzano, Italy; [d]Instituto Complutense de Estudios Internacionales, Universidad Complutense de Madrid, Madrid, Spain; [e]Department of Mathematics, Masjed-Soleiman Branch, Islamic Azad University, Masjed-Soleiman, Iran; [f]Department of Industrial Engineering, Eastern Mediterranean University, Famagusta, Turkey; [g]Department of Mathematics, Central Tehran Branch, Islamic Azad University, Tehran, Iran

ABSTRACT

We study a three-stage supply chain network (SCN) design problem for a single-product system. The SCN is composed of suppliers that provide raw materials for the plants, plants that produce and send the finished products to distribution centres (DCs), and DCs that transport finished products to the customers. The use of different conveyances and step-fixed costs improves the applicability and the results but increases the complexity, generating an *NP*-hard problem. The overall objective of the problem is to minimise the total cost, which is composed of the opening and transportation costs in all three stages. Two hybrid metaheuristics – genetic algorithm–variable neighbourhood search (GA-VNS) and variable neighbourhood search–simulated annealing (VNS-SA) – are proposed to solve this *NP*-hard problem. In addition to the novelty of the proposed algorithms, we develop an innovative priority-based decoding method to design chromosomes and solutions related to the nature of the problem. A robust parameter and operator setting is implemented using the Taguchi experimental design method with several random test problems. The performance of these algorithms is evaluated and compared for different problem sizes. The experimental results indicate that the GA-VNS is robust and superior to the other competing methods.

ARTICLE HISTORY
Received 27 July 2016
Accepted 2 April 2017

KEYWORDS
Supply chain network; multi-stage system; simulated annealing; hybrid metaheuristic algorithms; Taguchi experimental design

1. Introduction

Successful supply chain management is determined by sound decisions regarding the flow of information, products and funds in a multi-stage environment. A supply chain is often represented as a network of nodes describing facilities and arcs connecting nodes along with the products flowing through the network. Supply chain network (SCN) design plays a pivotal role in minimising the total cost of the supply chain. A multi-stage SCN (MSCN) is formed by sequencing multiple SCN stages for transferring flows between two successive stages. An MSCN generally includes suppliers, plants, distribution centres (DCs) and customers. The suppliers provide raw materials for the plants, the plants produce and send the finished products to the DCs, and the DCs transport the product to the customers. As many other engineering problems (see Alfares, 2015; Jablonsky, 2007; Kovács & Marian, 2002), this one is also optimised using mathematical formulations.

Several exact and heuristic approaches, such as the Lagrangian relaxation (LR) method, have been developed to solve a wide range of *NP*-hard problems, including MSCN design problems for different sectors (Alfares, 2015; Altiparmak, Gen, Lin, & Karaoglan, 2009; Lančinskas et al. 2015; Vizvári, Lakner, Csizmadia, & Kovács, 2011). For example, Jayaraman and Pirkul (2001) proposed an LR heuristic for solving single-source, multi-product, MSCN design problems. Syam (2002) applied the LR and simulated annealing (SA) methods to a multi-source, multi-product, multi-location framework. A hybrid model of network design and production/distribution planning for an SCN problem was presented by Jang, Jang, Chang, and Park (2002). They used a LR heuristic to design the SCN and developed a genetic algorithm (GA) to integrate the production and distribution planning problems. Amiri (2006) developed a mixed integer-programming model to minimise the total cost of a two-stage uncapacitated SCN distribution network. To solve the problem, he proposed an efficient heuristic solution based on the LR technique.

For large problem instances, the exact methods are computationally expensive. Because of their good performance and short run time, metaheuristics have been

CONTACT Madjid Tavana tavana@lasalle.edu
[†]Department of Industrial Engineering, Firouzabad Institute of Higher Education, Firouzabad, Fars, Iran

recently used to solve large-scale problems. Zhou, Min, and Gen (2002) were the first researchers who proposed using metaheuristics to solve SCN design problems. They modelled the problem as a balanced allocation of customers to multiple distribution centres and implemented a GA to solve it. Syarif, Yun, and Gen (2002) formulated the design of a multi-source, single-product MSCN as a mixed-integer linear programming problem. To solve the model, they proposed a novel spanning-tree-based GA based on Prüfer numbers. They also developed a repairing procedure to handle infeasible chromosomes.

Yeh (2005) modified the mathematical model defined by Syarif et al. (2002). He proposed an algorithm that employed a simple greedy and a hybrid local search method combining the exchange, remove and insert procedures. Later on, Yeh (2006) proposed a memetic algorithm for the problem, which combined a GA, the local search and greedy heuristic methods, and a linear programming approach. In order to evaluate the performance of the proposed algorithm, a GA and a heuristic method were also implemented and compared to the proposed memetic algorithm.

Altiparmak et al. (2009) considered a single-source, multi-product, MSCN design problem. They proposed a solution method based on a GA endowed with an encoding structure to represent a solution for the problem and greedy heuristics to generate the initial population. The effectiveness of their GA was investigated by comparing its results with those obtained by the CPLEX, LR, hybrid GA and SA methods. Costa, Celano, Fichera, and Trovato (2010) presented a new encoding/decoding procedure embedded within a GA to minimise the total cost in a single-product MSCN design problem. Their procedure allowed them to avoid the decoding of unfeasible distribution flows at the stage of the supply chain transporting products from plants to DCs.

Olivares-Benitez, Gonza´lez-Velarde, and Rı´os-Mercado (2012) analysed a single-product two-stage solid SCN design problem, where a product is distributed from plants to DCs and then to customers. They formalised this framework as a bi-objective (cost and time) minimisation problem. The authors proposed three variations of the classic epsilon-constraint method to generate Pareto fronts. Later on, Olivares-Benitez, Rı´os-Mercado, and Gonza´lez-Velarde (2013) developed a metaheuristic algorithm to solve the same problem. Their algorithm combined elements from greedy functions, Scatter Search, Path Relinking and Mathematical Programming. It decomposed the construction of a solution into a hierarchy of decisions. Large problem instances were solved using the metaheuristic algorithm, whose time and quality were compared with those of the epsilon-constraint method. The results favoured the metaheuristic algorithm for large instances of the problem.

Kristianto, Gunasekaran, Helo, and Hao (2014) developed an SCN by optimising inventory allocation and transportation routing. They incorporated a fuzzy shortest path algorithm into a two-stage programming problem in order to find the global optimum solution. Melo, Nickel, and Saldanha-da-Gama (2014) studied a multi-period logistics network redesign problem and proposed a two-phase LP-based heuristic method modelled as a large-scale mixed-integer linear program to solve it. Khalifehzadeh, Seifbarghy, and Naderi (2015) considered a four-echelon SCN design with shortage. They presented a multi-objective mathematical model to minimise the total operating costs of all the supply chain elements and to maximise the reliability of the system. They solved this problem using a comparative particle swarm optimisation algorithm.

In this paper, we consider a three-stage MSCN design problem with solid transportation. The use of different conveyances and step-fixed costs make the problem more realistic and, at the same time, more complex to solve. In other words, the novel framework studied in the current paper improves upon the recent two-stage (Hajiaghaei-Keshteli (2011), and two-stage solid SCN models, Molla-Alizadeh-Zavardehi, Sadi Nezhad, and Tavakkoli-Moghaddam (2013) and Olivares-Benitez et al. (2012, 2013), by adding a third stage to the design of the SCN while accounting for the conveyances used through the different stages of the network.

The problem is formulated both as a pure-integer nonlinear program and as a 0–1 mixed-integer linear program. To solve such an *NP*-hard problem, two hybrid metaheuristics – Genetic Algorithm–Variable Neighbourhood Search (GA-VNS) and Variable Neighbourhood Search–Simulated Annealing (VNS-SA) – are proposed.

In addition to the novelty of the proposed algorithms, the priority-based decoding method is developed to design chromosomes/solutions that are suited to the characteristics of our design problem. In particular, the priority-based encoding method applied to solid transportation problems (Altiparmak, Gen, Lin, & Paksoy, 2006) is extended and adapted to account for the effects of the step-fixed costs through the different stages of the SCN.

The remainder of the paper is organised as follows. In Section 2, we describe the problem and define the mathematical model. Section 3 presents the proposed solution methods. Section 4 describes the Taguchi experimental design method and compares the computational results obtained. Section 5 concludes and suggests some future research directions.

2. Problem description and mathematical model

The MSCN design problem with solid transportation defined through this section consists of suppliers, plants, DCs and customers. In the first stage, the suppliers provide raw materials for the plants. In the second stage, the plants produce and send the resulting product to DCs. Finally, the DCs transport the product to the customers. Conveyances, which can be considered as transportation types (truck, rail, airplane and ship), have different costs associated. One conveyance must be selected to transport the product in each stage.

The objective of the problem is the minimisation of the total supply chain costs such that all the capacity constraints and demand requirements of the customers are satisfied. The problem is formulated based on the following assumptions:

- We focus on the design of an MSCN with solid transportation and a single product.
- The number of suppliers, the maximum number of plants, the maximum number of DCs and conveyances, along with their capacities, and the number of customers, together with their demands, are all known.
- There is a step-fixed cost for each route in each stage.

Given these assumptions, the problem formulated aims at selecting a subset of plants and DCs that configure the best possible multi-stage distribution network flow through the different stages such that the demand requirements of the customers are fulfilled with the minimum cost. The following notations are used to define the mathematical model:

Set of indices:

S	Set of suppliers ($s = 1, 2, \ldots, S$)
I	Set of plants ($i = 1, 2, \ldots, I$)
J	Set of DCs ($j = 1, 2, \ldots, J$)
K	Set of customers ($k = 1, 2, \ldots, K$)
M	Set of conveyances in the first stage ($m = 1, 2, \ldots, M$)
N	Set of conveyances in the second stage ($n = 1, 2, \ldots, N$)
L	Set of conveyances in the third stage ($l = 1, 2, \ldots, L$)

Parameters:

cap_s	Capacity of supplier s
cp_i	Capacity of plant i
cd_j	Capacity of DC j
dc_k	Demand of customer k
u	Utilisation rate of raw material per unit of the product
fcp_i	Fixed cost of operating plant i
fcd_j	Fixed cost of operating DC j
a_{sim}	Cost of purchasing/transporting raw material from supplier s to plant i by conveyance m
b_{ijn}	Cost of transporting one unit of product from plant i to DC j by conveyance n
c_{jkl}	Cost of delivering one unit of product from DC j to customer k by conveyance l
$f_{sim,1}$	First fixed cost of transporting raw material from supplier s to plant i by conveyance m
$f_{sim,2}$	Second fixed cost of transporting raw material from supplier s to plant i by conveyance m
$g_{ijn,1}$	First fixed cost of transporting product from plant i to DC j by conveyance n
$g_{ijn,2}$	Second fixed cost of transporting product from plant i to DC j by conveyance n
$h_{jkl,1}$	First fixed cost of delivering product from DC j to customer k by conveyance l
$h_{jkl,2}$	Second fixed cost of delivering product from DC j to customer k by conveyance l
q_{sim}	Maximum quantity of raw material that can be transported from supplier s to plant i by conveyance m without incurring in the second fixed cost
o_{ijn}	Maximum quantity of product that can be transported from plant i to DC j by conveyance n without incurring in the second fixed cost
w_{jkl}	Maximum quantity of product that can be delivered from DC j to customer k by conveyance l without incurring in the second fixed cost
pp_i	Unit production cost of product at plant i
sd_j	Unit storing cost of product at DC j
$E_m^{(1)}$	Maximum capacity of conveyance m in the first stage
$E_n^{(2)}$	Maximum capacity of conveyance n in the second stage
$E_l^{(3)}$	Maximum capacity of conveyance l in the third stage

Decision variables:

P_i	Binary variable equal to 1 if plant i is opened and equal to 0 otherwise
D_j	Binary variable equal to 1 if DC j is opened and equal to 0 otherwise
X_{sim}	Quantity of raw material shipped from supplier s to plant i by conveyance m
Y_{ijn}	Quantity of product shipped from plant i to DC j by conveyance n
Z_{jkl}	Quantity of product shipped from DC j to customer k by conveyance l
$T_{sim}^{(1)}$	Binary variable equal to 1 if $X_{sim} > 0$ and equal to 0 otherwise
$T_{ijn}^{(2)}$	Binary variable equal to 1 if $Y_{ijn} > 0$ and equal to 0 otherwise
$T_{jkl}^{(3)}$	Binary variable equal to 1 if $Z_{jkl} > 0$ and equal to 0 otherwise

The mathematical model of the MSCN design problem with solid transportation in which the transportation costs are step fixed will be formulated as (1) a nonlinear programming model and (2) a 0–1 mixed-integer linear programming model.

2.1. Nonlinear programming model

The formulation of the problem provided in this section and its linear version described in the next follow the same intuition as the standard models on MSCN design, such as Mehdizadeh, Afrabandpei, Mohaselafshar, and Afshar-Nadja (2013), Molla-Alizadeh-Zavardehi et al. (2013) and Olivares-Benitez Mehdizadeh et al. (2013). That is, we formalise a cost minimisation problem subject to different capacity constraints on the side of the suppliers, plants, DCs and conveyances connecting all three stages while satisfying a given consumer demand:

$$\text{Min}Z = \sum_{s=1}^{S}\sum_{i=1}^{I}\sum_{m=1}^{M} a_{sim} \times X_{sim} + \sum_{i=1}^{I}\sum_{j=1}^{J}\sum_{n=1}^{N} b_{ijn} \times Y_{ijn} + \sum_{j=1}^{J}\sum_{k=1}^{K}\sum_{l=1}^{L} c_{jkl} \times Z_{jkl}$$

$$+\sum_{s=1}^{S}\sum_{i=1}^{I}\sum_{m=1}^{M}(\alpha_{sim,1}\times f_{sim,1}+\alpha_{sim,2}\times f_{sim,2})$$
$$+\sum_{i=1}^{I}\sum_{j=1}^{J}\sum_{n=1}^{N}(\beta_{ijn,1}\times g_{ijn,1}+\beta_{ijn,2}\times g_{ijn,2})$$
$$+\sum_{j=1}^{J}\sum_{k=1}^{K}\sum_{l=1}^{L}(\gamma_{jkl,1}\times h_{jkl,1}+\gamma_{jkl,2}\times h_{jkl,2})$$
$$+\sum_{i=1}^{I}fcp_i\times P_i+\sum_{j=1}^{J}fcd_j\times D_j$$
$$+\sum_{i=1}^{I}\sum_{j=1}^{J}\sum_{n=1}^{N}pp_i\times Y_{ijn}+\sum_{i=1}^{I}\sum_{j=1}^{J}\sum_{n=1}^{N}sd_j\times Y_{ijn} \quad (1)$$

s.t.

$$\sum_{i=1}^{I}\sum_{m=1}^{M}X_{sim}\le cap_s \qquad \forall s \quad (2)$$

$$\sum_{j=1}^{J}\sum_{n=1}^{N}Y_{ijn}\le cp_i\times P_i \qquad \forall i \quad (3)$$

$$\sum_{j=1}^{J}\sum_{n=1}^{N}u\times Y_{ijn}\le\sum_{s=1}^{S}\sum_{m=1}^{M}X_{sim} \qquad \forall i \quad (4)$$

$$\sum_{i=1}^{I}\sum_{n=1}^{N}Y_{ijn}\le cd_j\times D_j \qquad \forall j \quad (5)$$

$$\sum_{k=1}^{K}\sum_{l=1}^{L}Z_{jkl}\le\sum_{i=1}^{I}\sum_{n=1}^{N}Y_{ijn} \qquad \forall j \quad (6)$$

$$\sum_{j=1}^{J}\sum_{l=1}^{L}Z_{jkl}\ge dc_k \qquad \forall k \quad (7)$$

$$\sum_{s=1}^{S}\sum_{i=1}^{I}X_{\mathrm{sim}}\le E_m^{(1)} \qquad \forall m \quad (8)$$

$$\sum_{i=1}^{I}\sum_{j=1}^{J}Y_{ijn}\le E_n^{(2)} \qquad \forall n \quad (9)$$

$$\sum_{j=1}^{J}\sum_{k=1}^{K}Z_{jkl}\le E_l^{(3)} \qquad \forall l \quad (10)$$

$$\alpha_{sim,1}=\begin{cases}1 \ if X_{sim}>0\\ 0 \ \text{otherwise}\end{cases} \qquad \forall s,i,m \quad (11)$$

$$\alpha_{sim,2}=\begin{cases}1 \ if X_{sim}>q_{sim}\\ 0 \ \text{otherwise}\end{cases} \qquad \forall s,i,m \quad (12)$$

$$\beta_{ijn,1}=\begin{cases}1 \ if Y_{ijn}>0\\ 0 \ \text{otherwise}\end{cases} \qquad \forall i,j,n \quad (13)$$

$$\beta_{ijn,2}=\begin{cases}1 \ if Y_{ijn}>o_{ijn}\\ 0 \ \text{otherwise}\end{cases} \qquad \forall i,j,n \quad (14)$$

$$\gamma_{jkl,1}=\begin{cases}1 \ if Z_{jkl}>0\\ 0 \ \text{otherwise}\end{cases} \qquad \forall j,k,l \quad (15)$$

$$\gamma_{jkl,2}=\begin{cases}1 \ if Z_{jkl}>w_{jkl}\\ 0 \ \text{otherwise}\end{cases} \qquad \forall j,k,l \quad (16)$$

$$X_{sim},Y_{ijn},Z_{jkl}\ge 0 \qquad \forall s,i,j,k,m,n,l \quad (17)$$

In the model above, the objective function (1) minimises the total cost of the SCN, which includes the following terms:

- the variable cost (determined by the quantity) of transporting raw material from the suppliers to the plants
- the variable cost of transporting the products from the plants to the DCs
- the variable cost of delivering the products from the DCs to the customers
- the first and second step-fixed costs for the three previous stages depending on the conveyance chosen
- the fixed costs of operating the plants and the DCs
- the production cost of the product at the plant
- the storing cost of the product at the DC.

The following constraints are imposed:

Equation (2) limits the quantity of raw material shipped from the suppliers to their respective capacities; Equation (3) limits the quantity of product shipped from an operational plant to its capacity; Equation (4) describes the product flow conservation between suppliers and plants, i.e. the quantity of product that can be obtained by a plant from the raw material received given the utilisation rate; Equation (5) limits the quantity of product that can be shipped to an operational DCs to its corresponding capacity; Equation (6) states that the total quantity of product delivered by a DC to the different customers cannot exceed the amount of product shipped to the DC; Equation (7) requires the demand from each customer to be satisfied; Equations (8)–(10) limit the quantities transported to the capacity of the different conveyances used in the first, second and third stage, respectively; Equations (11), (13) and (15) ensure that an initial fixed cost is incurred if a shipment route carries a positive quantity; Equations (12), (14) and (16) ensure that an additional fixed cost is incurred if a certain shipment route exceeds the corresponding step-fixed quantity limit; Equation (17) imposes non-negative restrictions on the decision variables X_{sim}, Y_{ijn} and Z_{jkl}.

Given the total cost minimisation defined by the objective function, no extra raw material or product should be transported through the different stages of the supply chain in the optimal solution. Therefore,

constraints (4), (6) and (7) should hold with equality in the optimal solution.

2.2. 0–1 mixed-integer linear programming model

$$\begin{aligned}\text{Min}Z = &\sum_{s=1}^{S}\sum_{i=1}^{I}\sum_{m=1}^{M} a_{sim} \times X_{sim} + \sum_{i=1}^{I}\sum_{j=1}^{J}\sum_{n=1}^{N} b_{ijn} \times Y_{ijn} \\ &+ \sum_{j=1}^{J}\sum_{k=1}^{K}\sum_{l=1}^{L} c_{jkl} \times Z_{jkl} \\ &+ \sum_{s=1}^{S}\sum_{i=1}^{I}\sum_{m=1}^{M} (\alpha_{sim,1} \times f_{sim,1} + \alpha_{sim,2} \times f_{sim,2}) \\ &+ \sum_{i=1}^{I}\sum_{j=1}^{J}\sum_{n=1}^{N} (\beta_{ijn,1} \times g_{ijn,1} + \beta_{ijn,2} \times g_{ijn,2}) \\ &+ \sum_{j=1}^{J}\sum_{k=1}^{K}\sum_{l=1}^{L} (\gamma_{jkl,1} \times h_{jkl,1} + \gamma_{jkl,2} \times h_{jkl,2}) \\ &+ \sum_{i=1}^{I} fcp_i \times P_i + \sum_{j=1}^{J} fcd_j \times D_j \\ &+ \sum_{i=1}^{I}\sum_{j=1}^{J}\sum_{n=1}^{N} pp_i \times Y_{ijn} + \sum_{i=1}^{I}\sum_{j=1}^{J}\sum_{n=1}^{N} sd_j \times Y_{ijn} \end{aligned} \tag{18}$$

s.t.

$$\sum_{i=1}^{I}\sum_{m=1}^{M} X_{sim} \le cap_s \qquad \forall s \tag{19}$$

$$\sum_{j=1}^{J}\sum_{n=1}^{N} Y_{ijn} \le cp_i \times P_i \qquad \forall i \tag{20}$$

$$\sum_{j=1}^{J}\sum_{n=1}^{N} u \times Y_{ijn} \le \sum_{s=1}^{S}\sum_{m=1}^{M} X_{sim} \qquad \forall i \tag{21}$$

$$\sum_{i=1}^{I}\sum_{n=1}^{N} Y_{ijn} \le cd_j \times D_j \qquad \forall j \tag{22}$$

$$\sum_{k=1}^{K}\sum_{l=1}^{L} Z_{jkl} \le \sum_{i=1}^{I}\sum_{n=1}^{N} Y_{ijn} \qquad \forall j \tag{23}$$

$$\sum_{j=1}^{J}\sum_{l=1}^{L} Z_{jkl} \ge dc_k \qquad \forall k \tag{24}$$

$$\sum_{s=1}^{S}\sum_{i=1}^{I} X_{sim} \le E_m^{(1)} \qquad \forall m \tag{25}$$

$$\sum_{i=1}^{I}\sum_{j=1}^{J} Y_{ijn} \le E_n^{(2)} \qquad \forall n \tag{26}$$

$$\sum_{j=1}^{J}\sum_{k=1}^{K} Z_{jkl} \le E_l^{(3)} \qquad \forall l \tag{27}$$

$$X_{sim} \le M_{sim} \times \alpha_{sim,1} \qquad \forall s, i, m \tag{28}$$

$$X_{sim} - q_{sim} \le M_{sim} \times \alpha_{sim,2} \qquad \forall s, i, m \tag{29}$$

$$Y_{ijn} \le N_{ijn} \times \beta_{ijn,1} \qquad \forall i, j, n \tag{30}$$

$$Y_{ijn} - o_{ijn} \le N_{ijn} \times \beta_{ijn,2} \qquad \forall i, j, n \tag{31}$$

$$Z_{jkl} \le R_{jkl} \times \gamma_{jkl,1} \qquad \forall j, k, l \tag{32}$$

$$Z_{jkl} - w_{jkl} \le R_{jkl} \times \gamma_{jkl,1} \qquad \forall j, k, l \tag{33}$$

$$\alpha_{sim,1}, \alpha_{sim,2} \in \{0, 1\} \qquad \forall s, i, m \tag{34}$$

$$\beta_{ijn,1}, \beta_{ijn,2} \in \{0, 1\} \qquad \forall i, j, n \tag{35}$$

$$\gamma_{jkl,1}, \gamma_{jkl,2} \in \{0, 1\} \qquad \forall j, k, l \tag{36}$$

$$X_{sim}, Y_{ijn}, Z_{jkl} \ge 0 \qquad \forall s, i, j, k, m, n, l \tag{37}$$

where $M_{sim} = \min\{cap_s, cp_i, E_m^{(1)}\}$, $N_{ijn} = \min\{cp_i, cd_j, E_n^{(2)}\}$, $R_{jkl} = \min\{cd_j, dc_k, E_l^{(3)}\}$.

The objective function (18) together with constraints (19)–(27) are identical to their counterparts defined in Equations (1)–(10); (28), (30) and (32) are equivalent to (11), (13) and (15), respectively. These constraints state that a fixed cost is incurred if a shipment route carries a positive quantity. This quantity is limited by the demand of the customers together with the capacity of the suppliers, plants, DCs and conveyances involved in the shipping process through the different stages; similarly, Equations (29), (31) and (33) are equivalent to Equations (12), (14) and (16), respectively. These constraints state that an additional fixed cost is incurred if a certain shipment route exceeds the corresponding step-fixed quantity limit; Equations (34)–(36) are standard binary constraints, while Equation (37) imposes non-negative restrictions on the decision variables X_{sim}, Y_{ijn} and Z_{jkl}.

As was the case in the nonlinear setting, the total cost minimisation defined by the objective function implies that no extra raw material or product should be transported through the different stages of the supply chain in the optimal solution. Therefore, constraints (21), (23) and (24) should hold with equality in the optimal solution.

3. Solution methods

The MSCN design problem with solid transportation and step-fixed costs is *NP*-hard (Molla-Alizadeh-Zavardehi

et al., 2013). Therefore, instead of trying to find the optimal solution, many researchers have developed a wide range of heuristics and metaheuristics methods to achieve high-quality solutions within a reasonable time.

3.1. Representation

Representation is one of the most important issues affecting the performance of metaheuristic algorithms. Several representation methods for solving linear and nonlinear optimisation problems have been proposed in the literature by a number of researchers. In the following, five of the most well-known and frequently used methods are listed:

- Matrix-based representation (Michalewicz, Vignaux, & Hobbs, 1991)
- Spanning-tree-based representation by Prüfer numbers (Gen & Cheng, 2000)
- Direct transportation tree representation (Eckert & Gottlieb, 2002)
- Priority-based encoding (Gen et al. 1997, Gen, Altiparmak, & Lin, 2006)
- Basic feasible solution representation (Liu, Yang, Mu, & Jiao, 2008)

Gen et al. (2006) discussed some of the drawbacks of these methods (for a comprehensive discussion, see Lotfi & Tavakoli-Moghadam, 2013) and proposed a new representation method built on priority-based encoding for single-product two-stage transportation problems. Priority-based encoding belongs to the permutation encoding class and it needs no repairing mechanism (Lotfi & Tavakoli-Moghadam, 2013). It has been successfully applied to the shortest path and project scheduling problems by Gen et al. (2000), multi-stage logistics network design by Lin, Gen, and Wang (2009), single-product multi-objective optimisation of SCN by Altiparmak et al. (2006), reverse logistics network design by Lee, Gen, and Rhee (2009), multi-product SCN design by Altiparmak et al. (2009) and the fixed-charge transportation problem by Lotfi et al. (2013). We apply priority-based encoding to the single-stage Step-Fixed Cost Solid Transportation Problem (SFCSTP) and the design of MSCNs with solid transportation. Some of the reasons for using priority-based encoding are (1) the implementation of this method is easy and its computational burden low; (2) it uses a transportation cost matrix in the decoding process.

In priority-based encoding, a digit in a solution contains two kinds of information: the position of the digit within the structure of the chromosome and its value. For example, in a transportation problem, the position of a digit is used to represent a node (source/depot in the transportation network) and its value is used to represent the priority of the corresponding node for constructing a tree among the different candidates (Altiparmak et al., 2006).

When priority-based encoding is applied to the single-stage SFCSTP, a solution consists of priorities of sources, depots and conveyances. In this case, the solution length in each stage is equal to the total number of sources (M), depots (N) and conveyances (P), i.e. $|M| + |N| + |P|$. The initial values are randomly generated from a permutation of digits ranging from 1 to $|M| + |N| + |P|$. To obtain the priority-based encoding for any SFCSTP, a priority must be initially assigned to the node with the highest value, $(|M| + |N| + |P|)$, and then we must proceed by sequentially reducing one until all the nodes have been assigned a priority. In the SFCSTP, we use the allocation matrix, which is a three-dimensional array. The procedure followed to decode the solution of the SFCSTP is described in Figure 1.

Consider a simple example of the SFCSTP with two plants $A = (a_1 = 150,\ a_2 = 100)$, three depots $B = (b_1 = 70,\ b_2 = 50,\ b_3 = 60)$, two conveyances $C = (e_1 = 100,\ e_2 = 80)$ and total demand of 150. The remaining information is provided in Table 1. Table 2 presents the trace table of the decoding procedure for the solution [2 6|1 5 4| 3 7] defined in Iteration 0.

In the MSCN design problem with solid transportation, a solution consists of three segments. Each segment is used to obtain a product flow for a given stage, i.e. the rth segment of a solution matches the rth stage of the MSCN. Therefore, the length of the solution is equal to the total length of the first segment, $(S + I + M)$, plus the length of the second, $(I + J + N)$, and the third, $(J + K + L)$. That is, the total length of the solution is equal to $S + 2 \times (I + J) + K + M + N + L$. The overall decoding procedure applied to the priority-based encoding of the three-stage SCN design problem with solid transportation and step-fixed costs is presented in Figure 2. Figures 3–5 provide the decoding procedures for the third, second and first stages, respectively. Note that the SFCSTP is recalled in each one of these stages, accounting for the effect of the step-fixed costs on the design process.

3.2. Genetic algorithm

A GA is a directed random search method based on the principles of evolution theory and used as a robust optimisation technique to solve many real-world problems (Gen & Cheng, 1997, 2000; Sanz-García, Pernía-Espinoza, Fernández-Martínez, & Martínez-de-Pisón-Ascacíbar, 2012; Souza dos Santos & Silva Formiga, 2014;

Procedure 1

Input:

m: Number of sources, n: number of depots, p: number of conveyances, K: number of customers,

a_i : Supply of source i, i=1,2,…,m, b_j : demand of depot j, j=1,2,…,n,

e_k : Capacity of conveyance k, k=1,2,…,p, dc_k : demand of customer k, k = 1,2,…,K.

TD: Total remaining demand of costumers, which is equal to $\sum_{k=1}^{K} dc_k$.

c_{ijk} : Variable transportation cost of one unit of product from source i to depot j by conveyance k.

A_{ijk} : Maximum quantity of product that can be sent from plant i to DC j with conveyance k without incurring in the second fixed cost, $\forall i,j,k$.

$f_{ijk,1}$: Fixed transportation cost associated with route *(i,j,k)*.

$f_{ijk,2}$: Additional fixed cost for the excess shipment from A_{ijk}, $v(i+j+k)$: Solution/chromosome.

Output:

X_{ijk} : The amount of product transported from source i to depot j by conveyance k,

Step 1:

$x_{ijk} \leftarrow 0$, for each i ,j, k,

Step 2:

$q \leftarrow \arg\max\{v(t): t = 1,2,...,m+n+p\}$: select a node,

Step 3:

If $q \le m$, then $i^* \leftarrow q$: select a source,

$$(j^*,k^*) \leftarrow \arg\min\{C_{i^*jk} = c_{i^*jk} + \frac{(f_{i^*jk,1} + f_{i^*jk,2})}{\min\{a_{i^*}, b_j, e_k\}} \mid v(m+j) \neq 0, j = 1,....,n, v(m+n+k) \neq 0,\ k = 1,....,p\}:$$

Select a depot and a conveyance with lowest cost, Else if $m < q \le m+n$, then $j^* \leftarrow q - m$: Select a depot,

$$(i^*,k^*) \leftarrow \arg\min\{C_{ij^*k} = c_{ij^*k} + \frac{(f_{ij^*k,1} + f_{ij^*k,2})}{\min\{a_i, b_{j^*}, e_k\}} \mid v(i) \neq 0, i = 1,....,m, v(m+n+k) \neq 0,\ k = 1,....,p\}:$$

Select a source and a conveyance with lowest cost, Else $k^* \leftarrow q - m - n$: Select a conveyance,

$$(i^*,j^*) \leftarrow \arg\min\{C_{ijk^*} = c_{ijk^*} + \frac{(f_{ijk^*,1} + f_{ijk^*,2})}{\min\{a_i, b_j, e_{k^*}\}} \mid v(i) \neq 0, i = 1,....,m, v(m+j) \neq 0,\ j = 1,....,n\}:$$

Select a source and a depot with lowest cost,

Step 4:

$x_{i^*j^*k^*} \leftarrow \min\{a_{i^*}, b_{j^*}, e_{k^*}, \mathrm{TD}\}$: assign available units,

Step 5:

$a_{i^*} \leftarrow a_{i^*} - x_{i^*j^*k^*}, b_{j^*} \leftarrow b_{j^*} - x_{i^*j^*k^*}, e_{k^*} \leftarrow e_{k^*} - x_{i^*j^*k^*}, \mathrm{TD} = \mathrm{TD} - x_{i^*j^*k^*}$ update availability of source i^*, depot j^* and conveyance k^*,

Step 6:

If $a_{i^*} = 0$ then $v(i^*) \leftarrow 0$, if $b_{i^*} = 0$ then $v(m+j^*) \leftarrow 0$, if $e_{k^*} = 0$ then $v(m+n+k^*) \leftarrow 0$: Remove priority of selected source, depot and conveyance,

Step 7:

While TD > 0, go to step 2, else calculate the total transportation cost.

Figure 1. Decoding procedure for the SFCSTP solution.

Table 1. Transportation cost and step-fixed cost for the simple example.

	c_{ijk}			$f_{ijk,1}$			$f_{ijk,2}$	
k	1	2	k	1	2	k	1	2
c_{11k}	3	5	$f_{11k,1}$	5	8	$f_{11k,2}$	3	8
c_{12k}	7	6	$f_{12k,1}$	10	20	$f_{12k,2}$	15	20
c_{13k}	5	4	$f_{13k,1}$	14	5	$f_{13k,2}$	5	9
c_{21k}	9	2	$f_{21k,1}$	3	9	$f_{21k,2}$	6	8
c_{22k}	1	9	$f_{22k,1}$	10	20	$f_{22k,2}$	10	14
c_{23k}	10	1	$f_{23k,1}$	4	17	$f_{23k,2}$	12	7

Stankiewicz, Roszak, & Morzyński, 2011). GAs work by generating a population of chromosomes, each representing a possible solution for a problem. The individuals in the population are evaluated and new solutions generated. These new chromosomes are created by reproduction, crossover or mutation. In other words, these three genetic operators take the initial population and generate successive populations that keep improving over time. The new populations are evaluated again and the whole process repeated. The overall procedure describing a standard GA is shown in Figure 6.

3.2.1. Selection

The main objective of the selection mechanism is to drive the search towards better individuals. By using the Roulette wheel selection mechanism, a solution with a higher fitness value has a higher chance of being selected for the new generation.

3.2.2. Reproduction

Better parents usually generate better offspring. Therefore, a percentage p_r of the *chromosomes* with a better fitness values is copied to the next generation.

3.2.3. Crossover

Crossover is the exchange of genes between the *chromosomes* of two parents. Since a percentage p_r of the best *chromosomes* is copied to the new generation, the remaining percentage $(1 - p_r)$ is produced through crossover. Commonly used operators are One-point, Two-point and Uniform crossover.

3.2.4. Mutation

Mutation is used to avoid convergence to a local optimum and to achieve a diverse population. Commonly used mutation operators are Swap, Big Swap, Inversion, Displacement and Perturbation.

3.3. Simulated annealing algorithm

SA is a probabilistic metaheuristic algorithm for finding the global optimum in a large search space. It was initially proposed by Kirkpatrick, Gelatt Jr., and Vecchi (1983) and has been successfully used in the optimisation of combinatorial problems (Che, 2012).

SA requires an initial solution s to be randomly generated and a random neighbour solution s' to be selected

Table 2. Trace table of the decoding procedure for the SFCSTP.

Iteration	$v(i+j+k)$	A	B	C	TD	C_{ijk}	i	j	k	x_{ijk}
0	[2 6|1 5 4| 3 7]	(150,100)	(70,50,60)	(100,80)	150	(5.32, 6.8, 4.28, 2.34, 9.02, 1.48)	2	3	2	60
1	[2 6|1 5 0| 3 7]	(150,40)	(70,50, 0)	(100,20)	90	(5.8, 8, –, 2.85, 10.7, –)	2	1	2	20
2	[2 6|1 5 0| 3 0]	(150,20)	(50,50,0)	(100, 0)	70	(3.16, 7.5, –, 9.45, 2.7, –)	2	2	1	20
3	[2 0|1 5 0| 3 0]	(150, 0)	(50,30,0)	(80,0)	50	(3.16, 7.83, –, –, –, –)	1	1	1	50

Procedure 2

Step 1: find Z_{jkl} by procedure 3 (third stage decoding);

Step 2: find Y_{ijn} by procedure 4 (secon stage decoding);

Step 3: find X_{sim} by procedure 5 (first stage decoding);

Step 4: calculate the value of objective function (Z) and **stop**.

Figure 2. Overall decoding procedure for the priority-based encoding of the three-stage SCN problem.

Procedure 3

Input:

J: Number of DCs, K: number of customers, L: number of conveyances,

cd_j : Capacity of DC j, $\forall j \in J$,

dc_k : Demand of costumer k, $\forall k \in K$,

$E_l^{(3)}$: Capacity of conveyance l, $\forall l \in L$,

c_{jkl} : Shipping cost of one unit of product from DC j to customer k by conveyance l,

$v_3(j+k+l)$: Solution, $\forall j \in J, \forall k \in K, \forall l \in L$,

Output:

Z_{jkl} : The amount of shipment from DC j to customer k by conveyance l,

cd'_j : Total customer demand from DC j, $\forall j \in J$,

Step 1:

Set DCs as sources, customers as depots and conveyances as conveyances, call procedure 1 to obtain $Z_{jkl}, a_i = cd_j, b_j = dc_k, e_k = E_l^{(3)}$.

Calculate cd'_j considering Z_{jkl} and **Return**.

Figure 3. Third-stage decoding procedure.

around it. Then, the change in the value of the objective function, $\Delta_{s,s'} = f(s') - f(s)$, is calculated. If $\Delta_{s,s'} \leq 0$, then solution s' replaces s. Otherwise, s' is accepted as the new solution for the next iteration with probability $p = \exp(-\Delta_{s,s'}/T)$, where T is a control parameter called temperature. A $n_{\max}$ fixed number of neighbourhood searches is performed for each temperature, which is gradually reduced in each iteration using a cooling schedule. This process is repeated until a termination condition is met.

Similarly to the mutation process of the GA, five commonly used neighbourhood operators are Swap, Big Swap, Inversion, Displacement and Perturbation. A general outline of the SA algorithm is provided in Figure 7.

Procedure 4

Input:

I: Number of plants, J: number of DCs, N: number of conveyances,

cp_i : Capacity of plant i, $\forall i \in I$,

cd_j ': Total customer demand for the product of DC j, $\forall j \in J$,

$E_n^{(2)}$: Capacity of conveyance n, $\forall n \in N$,

b_{ijn} : Shipping cost of one unit of product from plant i to DC j by conveyance n,

$v_2(i+j+n)$: Solution, $\forall i \in I, \forall j \in J, \forall n \in N$,

Output:

Y_{ijn} : The amount of shipment from plant i to DC j by conveyance n,

cp_i ': Total customer demand for the product of plant i, $\forall i \in I$,

Step 1:

Set plants as sources, DCs as depots and conveyances as conveyances, call procedure 1 to obtain $Y_{ijn}, a_i = cp_i, b_j = cd_j', e_k = E_n^{(2)}$.

Calculate cp_i' considering Y_{ijn} and **Return**.

Figure 4. Second-stage decoding procedure.

Procedure 5

Input:

S: Number of suppliers, I: number of plants, M: number of conveyances,

cap_s : Capacity of supplier s for raw material,

cp'_i : Total customer demand for the product of plant i, $\forall i \in I$,

$E_m^{(1)}$: Capacity of conveyance m, $\forall m \in M$,

a_{sim} : Cost of purchasing and transporting raw material from supplier s to plant i by conveyance m,

u : Utilization rate of raw material per unit of product,

a_i : Amount of raw material required to produce the product of plant i,

$v_1(s+i+m)$: solution, $\forall s \in S, \forall i \in I, \forall m \in M$,

Output:

X_{sim} : Amount of raw material shipped from supplier s to plant i by conveyance m, $\forall s \in S, i \in I, m \in M$,

Step 1:

$X_{sim} \leftarrow 0, \forall s \in S, \forall i \in I, \forall m \in M$,

Calculate the amount of raw material reqiured to produce the product of plant i, $a_i = cp'_i \times u, \forall i \in I$,

cp'_i : Total customer demand for the product of plant i, $\forall i \in I$,

Step 2:

Set suppliers as sources, plants as depots and conveyances as conveyances, call procedure 1 to obtain X_{sim}, $a_i = cap_s, b_j = cp'_i, e_k = E_m^{(1)}$, and **Return**.

Figure 5. First-stage decoding procedure.

3.4. Variable neighbourhood search algorithm

VNS, proposed by Mladenovic and Hansen (1997), is an effective metaheuristic method that has been extensively used in solving a variety of combinatorial optimisation problems. It is an algorithm based on systematic neighbourhood changes through the search process so as to avoid getting stuck in the local optimums. Figure 8 describes the main steps of the VNS algorithm.

In particular, this algorithm consists of three main steps: shaking, local search and move. VNS starts by generating an initial solution x. Then, a new solution x' is randomly produced through the shaking phase $N_{k=1}^{s}$ in the kth preselected neighbourhood structure. The solution is regenerated $n_{\max}$ times through a local search phase aimed at finding local optima x'' from the input solution x' (Eskandarpour, Hessameddin Zegordi, & Nikbakhsh, 2013).

GA algorithm steps

Parameter setting: Set Population size (*popsize*), Reproduction percentage (p_r), Mutation probability (p_m), and Stopping criterion.
Operator Selection: Select Crossover and Mutation operators.
Initialization.
Fitness evaluation.
X_{best} = Update
Repeat
Reproduction (p_r)
Crossover ($1-p_r$)
Mutation (All offspring produced by crossover, p_m)
Fitness evaluation
X_{best} = Update
Until the stopping criterion is met

Figure 6. GA algorithm steps.

SA algorithm steps

Initialization: Select an initial solution (s_0), an initial temperature (T_0), the number of neighborhood searches in each temperature $n_{\max}$, and termination.
Set $T \leftarrow T_0$ and $s \leftarrow s_0$;
Repeat
  Repeat
    Randomly select $s' \in N(s)$;
    Calculate $\Delta_{s,s'} = f(s') - f(s)$
    if $\Delta_{s,s'} \leq 0$ then $s \leftarrow s'$;
    else generate random R uniformly in the range (0, 1);
    if $R \leq \exp(-\Delta_{s,s'} / T_n)$ then $s \leftarrow s'$;
  Until iteration_ counter = nt
  Decrease of the temperature T;
Until the stopping criterion is met

Figure 7. SA algorithm steps.

VNS algorithm steps

Select the set of neighborhood structures N_k^s, for $k = 1,\ldots, k_{max}$, that will be used in the shaking step, and the set of neighborhood structures N_l^{ls}, for $l = 1,\ldots, l_{\max}$, that will be used in the local search step.
Number of iterations in each local search $n_{\max}$.
Find the initial solution x.
Repeat (external loop)
  Set $k \leftarrow 1$ and $l \leftarrow 1$;
  Repeat (internal loop)
    Shaking:
    Generate random point $x' \in N_k^s(x)$;
    Local search:
    Get solution, x';
    Set $n \leftarrow 1$;
    $x'' \leftarrow x'$;
    While $n \leq n_{\max}$ do
      $x_{new} \in N_l^{ls}(x'')$;
      if $f(x_{new}) \leq f(x'')$
        $x'' \leftarrow x_{new}$, $n \leftarrow n + 1$;
      end
    end While
    If $f(x'') \leq f(x)$
      $x \leftarrow x''$, $k \leftarrow 1$ and $l \leftarrow 1$;
      else
        $k \leftarrow k + 1$ and $l \leftarrow l + 1$;
    End
  Until $k > k_{\max}$ ($k_{\max} = 3$)
Until the stopping criterion is met

Figure 8. VNS algorithm steps.

The (local) optimal solution x'' obtained from the local search is compared with the initial solution x. If x'' is better than the initial solution x, then the search procedure continues with the kth neighbourhood search operator $N^s_{k=1}$. Otherwise, the $(k+1)$th neighbourhood search operator N^s_{k+1} is applied. This procedure is repeated until the maximum preselected number of neighbourhood structures, $k = k_{\max}$, is reached. Then, the iteration number is increased by one and the next iteration begins.

3.5. Hybrid metaheuristics

The main purpose of using hybrid metaheuristic approaches is to combine the advantages of two different algorithms since a good balance between exploration and exploitation constitutes an important property of any search process. Moreover, algorithms should have the minimum possible number of control parameters to tune. In this regard and unlike many other algorithms, the basic schemes of the VNS and its extensions are simple and require few parameters – or even none – to implement the algorithm.

Generally, metaheuristics can be classified into methods that provide a single solution versus those that perform a population-based search. Thus, the VNS can be hybridised with a single-solution-based algorithm or with a population-based one. The first alternative consists of inserting the SA algorithm into the VNS so that it can take over the local search process, improving the capacity to search the neighbourhood of a solution and addressing the problem of how to escape from the local optima. Furthermore, embedding the SA algorithm into the VNS mitigates the sensitivity of the latter to the initial search region and allows to explore a larger range of the solution space.

This brings us to the second alternative, given the fact that GAs are capable of exploring the whole solution space. However, GAs are relatively poor at finding the precise local optimum in the region where the algorithm converges. Consequently, GAs usually require more computing efforts to locate the optimal solution in the region of convergence (Wu, Yu, Jin, Lin, & Schiavone, 2004). In this regard, the VNS algorithm can be used to improve and speed up the local search capability of GAs. Thus, combining the solution space search capacity of GAs and the local search one of the VNS constitutes an intuitively promising approach.

In particular, the GA-VNS hybrid benefits from the search capacity of GA as the main algorithm while applying the VNS procedure to improve the individuals of the population. Ideally, GAs should be restricted to focus on the valid individuals composing a given population. However, imposing the necessary restrictions can lead to biases in the search process, requiring the implementation of continuous revisions and repair mechanisms. In this case, the VNS can be used as a local search mechanism that allows the GA to generate subsets of (valid) offspring by one-point crossover and swap mutation operators so as to seek for improved solutions without having to revise the process and implement repair mechanisms.

Table 3. Test problems characteristics.

Problem size $S \times M \times I \times N \times J \times L \times K$	Total E_s	Total D_i	Total W_j	Total demand	Total $E^{(1)}_m$	Total $E^{(2)}_n$	Total $E^{(3)}_l$	q_{sim}, o_{ijn}, w_{jkl}
$5 \times 2 \times 3 \times 2 \times 5 \times 2 \times 10$	3000	2000	1300	1000	2500	1500	1200	U(100, 300)
$10 \times 2 \times 5 \times 2 \times 10 \times 2 \times 20$	6000	4000	3000	2000	5000	3000	3000	U(80, 200)
$15 \times 2 \times 8 \times 2 \times 15 \times 2 \times 30$	8000	6000	5000	3000	7500	5500	4500	U(100, 300)
$20 \times 2 \times 10 \times 2 \times 20 \times 3 \times 40$	11,000	9000	7000	5500	9000	8000	6000	U(100, 300)
$25 \times 2 \times 15 \times 3 \times 25 \times 3 \times 45$	14,000	10,000	8000	5000	11,000	8000	7000	U(90, 300)
$30 \times 2 \times 50 \times 3 \times 30 \times 3 \times 50$	15,000	12,000	10,000	7000	13,000	10,000	9500	U(50, 200)
$35 \times 3 \times 60 \times 3 \times 35 \times 4 \times 60$	18,000	15,000	10,000	9000	18,000	9000	9000	U(100, 300)
$40 \times 3 \times 70 \times 3 \times 45 \times 4 \times 75$	20,000	18,000	15,000	11,000	18,000	15,000	13,000	U(90, 200)
$45 \times 3 \times 80 \times 4 \times 45 \times 4 \times 80$	22,000	20,000	16,000	13,000	20,000	18,000	13,000	U(70, 300)
$50 \times 3 \times 100 \times 5 \times 50 \times 4 \times 100$	25,000	23,000	20,000	15,000	24,000	20,000	18,500	U(100, 200)

Problem size	Problem type	Range of variable costs				Range of first and second fixed costs			
$S \times M \times I \times N \times J \times L \times K$		b	c-b	b-a	d-c	b	c-b	b-a	d-c
$5 \times 2 \times 3 \times 2 \times 5 \times 2 \times 10$	A	U(3, 7)	U(0, 1)	U(0.25, 1)	U(0.25, 1)	U(100, 400)	U(0, 50)	U(10, 50)	U(10, 50)
$10 \times 2 \times 5 \times 2 \times 10 \times 2 \times 20$	B	U(3, 7)	U(0, 1)	U(0.25, 1)	U(0.25, 1)	U(200, 800)	U(0, 100)	U(20, 100)	U(20, 100)
$15 \times 2 \times 8 \times 2 \times 15 \times 2 \times 30$	C	U(3, 7)	U(0, 1)	U(0.25, 1)	U(0.25, 1)	U(400, 1600)	U(0, 200)	U(40, 200)	U(40, 200)
$20 \times 2 \times 10 \times 2 \times 20 \times 3 \times 40$	D	U(3, 7)	U(0, 1)	U(0.25, 1)	U(0.25, 1)	U(800, 3200)	U(0, 400)	U(80, 400)	U(80, 400)
$25 \times 2 \times 15 \times 3 \times 25 \times 3 \times 45$									
$30 \times 2 \times 50 \times 3 \times 30 \times 3 \times 50$									
$35 \times 3 \times 60 \times 3 \times 35 \times 4 \times 60$									
$40 \times 3 \times 70 \times 3 \times 45 \times 4 \times 75$									
$45 \times 3 \times 80 \times 4 \times 45 \times 4 \times 80$									
$50 \times 3 \times 100 \times 5 \times 50 \times 4 \times 100$									

Table 4. Factors and their levels.

GA and VNS factors	GA levels	VNS levels	GA-VNS levels	SA factors	SA levels	VNS-SA levels
Pop size	*A*(1)-50		*A*(1)-40	T_0	*A*(1)-6000	*A*(1)-700
	A(2)-55		*A*(2)-45		*A*(2)-6500	*A*(2)-750
	A(3)-60		*A*(3)-50		*A*(3)-7000	*A*(3)-800
p_c	*B*(1)-70%		*B*(1)-85%	$n_{\max}$	*B*(1)-300	*B*(1)-40
	B(2)-75%		*B*(2)-90%		*B*(2)-350	*B*(2)-45
	B(3)-80%		*B*(3)-95%		*B*(3)-400	*B*(3)-50
p_m	*C*(1)-0.10		*C*(1)-0.2	α	*C*(1)-0.92	*C*(1)-0.89
	C(2)-0.15		*C*(2)-0.25		*C*(2)-0.91	*C*(2)-0.9
	C(3)-0.20		*C*(3)-0.3		*C*(3)-0. 9	*C*(3)-0.91
Crossover	*D*(1)-One-point		*D*(1)-One-point			
	D(2)-Two-point		*D*(2)-Two-point			
	D(3)-Uniform		*D*(3)-Uniform			
Mutation	*E*(1)-Swap		*E*(1)-Swap			
	E(2)-Big Swap		*E*(2)-Big Swap			
	E(3)-Inversion		*E*(3)-Inversion			
	E(4)-Displacement		*E*(4)-Displacement			
	E(5)-Perturbation		*E*(5)-Perturbation			
$n_{\max}$		*A*(1)-200	*F*(1)-25			
		A(2)-250	*F*(2)-30			
		A(3)-300	*F*(3)-35			

4. Experimental design

4.1. Instances

In order to evaluate the performance of the proposed algorithms, an experimental design is used to generate test data. For the sake of clarity, the data generated are presented in Table 3.

The data required to define a problem consist of the number of suppliers, plants, DCs, customers and conveyances in each stage. Moreover, the following parameters, both deterministic and stochastic, must also be defined: total capacity of suppliers (E_s), plants (D_i) and DCs (W_j); total demand of customers; total capacity of the conveyances used in each stage; the range of the variable and step-fixed costs together with the quantity limits leading suppliers, plants and DCs to incur in the second fixed costs.

In order to run the algorithms, we have randomly generated 40 problem sets with 10 different sizes. The size of the problems is determined by the number of suppliers, plants, DCs, customers and conveyances in each stage. For each size, four different types of problems, *A*, *B*, *C* and *D*, are considered. These types differ in the range of the first and second fixed costs applied in each stage, which is incremented following the alphabetical order of the problem type. The variable and fixed costs, together with their corresponding quantity limits, are generated using uniform distributions whose domains depend on the type of cost being considered.

4.2. Parameter setting

The performance of the metaheuristic algorithms depends on the proper selection of parameters and operators (Hajiaghaei-Keshteli, Molla-Alizadeh-Zavardehi, & Tavakkolimoghaddam, 2010). The control factors of the proposed algorithms are given by:

- For the GA: population size, crossover percentage (p_c), mutation probability (p_m), type of crossover and type of mutation;
- For the SA: initial temperature (T_0), number of neighbourhood searches in each temperature ($n_{\max}$) and reduction ratio of temperature (α).

The parameters and operators of the proposed algorithms and their levels are described in Table 4.

As explained above, we generate 40 test problems for the GA with 4 three-level factors (Pop size, p_c, p_m and crossover) and 1 five-level factor (Mutation). Moreover, due to their stochastic nature, each test problem is run 10 times. Therefore, the total number of runs performed for the GA problems is $40 \times 3^4 \times 5^1 \times 10$, which is equal to 162,000. Similarly, the total number of runs performed for the SA, VNS, hybrid GA-VNS and hybrid VNS-SA problems is equal to 10,800, 1200, 486,000 and 10,800, respectively.

Given the size of the resulting problems, the Taguchi experimental design method can be implemented to reduce the number of experiments and maintain robustness. Taguchi developed a transformation of the repetition data into the signal-to-noise (S/N) ratio, which provides a robust measure of variation (Phadke, 1989). The term 'signal' refers to the desirable value (mean response variable) while 'noise' describes the undesirable one (standard deviation). That is, the S/N ratio describes the amount of variation that is present in the response variable. The aim is to maximise the S/N ratio, which in the current problems is

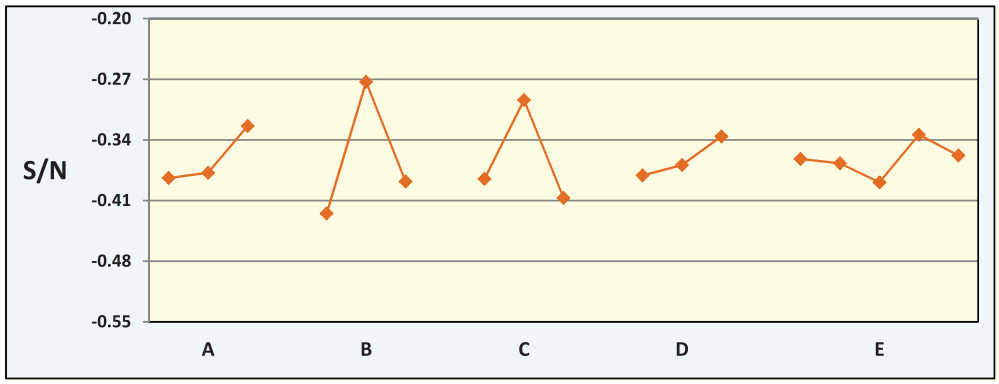

Figure 9. Mean S/N ratio plot for each level of the factors in GA.

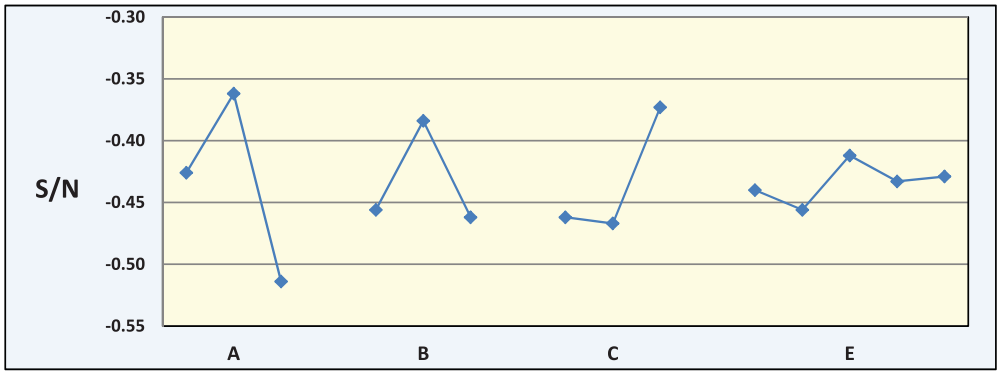

Figure 10. Mean S/N ratio plot for each level of the factors in SA.

defined as

$$\text{S/N ratio} = -10\log_{10}\left(\text{objective function}\right)^2$$

In order to select the most suitable orthogonal vector (array) for the GA, we need to calculate the total degrees of freedom. The array must have one degree of freedom related to the total mean, two degrees of freedom per each three-level factor (a total of $2 \times 4 = 8$ degrees of freedom) and four degrees of freedom for the five-level factor. Therefore, the total degrees of freedom required would be 13. Consequently, the suitable array should have at least 13 rows.

The resulting orthogonal array has to accommodate the different combinations of the factor levels in the experiment. Using the predesigned orthogonal arrays, which are given as a standard table, the L_{18} array described in Table 5 has been selected as a suitable one satisfying all of the design requirements. In this table, the columns represent the control factors and each row describes an experiment with its related combination of factors' levels. The experiments on the parameters of the GA have been generated according to the L_{18} array, yielding 18 combinations of control factors.

Table 5. The modified orthogonal array L18 for GA.

Trial	*A*	*B*	*C*	*D*	*E*
1	1	1	1	1	1
2	1	1	2	2	2
3	1	2	1	3	3
4	1	2	3	1	4
5	1	3	2	3	5
6	1	3	3	2	5
7	2	1	1	3	5
8	2	1	3	1	5
9	2	2	2	2	1
10	2	2	3	3	2
11	2	3	1	2	4
12	2	3	2	1	3
13	3	1	2	3	4
14	3	1	3	2	3
15	3	2	1	2	5
16	3	2	2	1	5
17	3	3	1	1	2
18	3	3	3	3	1

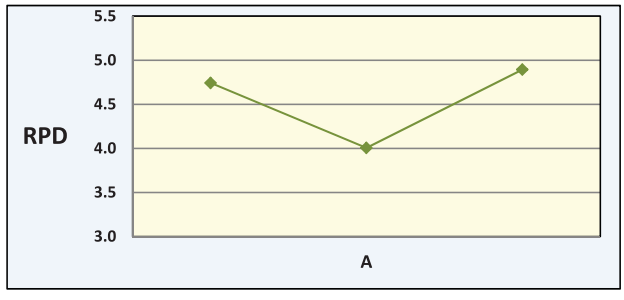

Figure 11. Mean RPD ratio plot for each level of the factors in VNS.

The suitable orthogonal array for the SA problem has been chosen using the same procedure. Due to the small number of runs in the VNS algorithm, the full factorial design method has been employed to measure the performance of each instance. In particular, VNS performance has been measured using the relative percentage deviation (RPD)

$$\text{RPD} = \frac{\text{Max}_{\text{sol}} - \text{Alg}_{\text{sol}}}{\text{Max}_{\text{sol}}} \times 100$$

where Alg_{sol} is the objective value obtained for a given instance and Max_{sol} is the best known solution for each instance.

Forty problems of different sizes have been generated and solved to study the performance of the algorithms. Because of the randomness and stochastic behaviour of the algorithms, each test problem has been run 10 times in order to obtain reliable results. After obtaining the results of the test problems in different trials, the results of each trial have been transformed into S/N ratios. The S/N ratios of the trials have been averaged for each level, and their values plotted against each control factor. It should be reminded that a higher value of the S/N ratio indicates a more robust algorithm.

Figure 9 provides the results for the GA, whose robustness is increased when the parameters of the factors *A*, *B*, *C*, *D*, and *E* are "defined as" 3, 2, 2, 3, and 4, respectively.

As can be seen in Figure 10, the best level of the parameters for the SA is set to 2, 2, 3 and 3, respectively. Also, $n_{\max}$ corresponds to level 2 in the VNS algorithm, as shown in Figure 11.

Figure 12 illustrates the best levels for the hybrid GA-VNS algorithm, which are defined as 1, 2, 2, 3, 1 and 4, respectively. Finally, in conformity with Figure 13, the best parameters for the proposed VNS-SA algorithm are given by 2, 2 and 1, respectively.

4.3. Experimental results

For comparison purposes, we require all the algorithms to run for an identical time of $0.6 \times (S + 2 \times (I + J) + K + M + N + L)$ seconds. Note that this criterion is determined by all the characteristics defining the problem. In other words, any increase in the size of the problem leads to a direct increment of the search time. Moreover, we generate 20 instances for each of the 10 problem sizes described in Table 3 and each instance is run 10 times. That is, a total of 200 instances, different from the test problems generated for calibration purposes, are produced per algorithm and problem size.

In order to measure the robustness of the algorithms, we analyse the effects that different problem sizes have on their relative performances. In particular, the reciprocal between the capability of the algorithms and the size

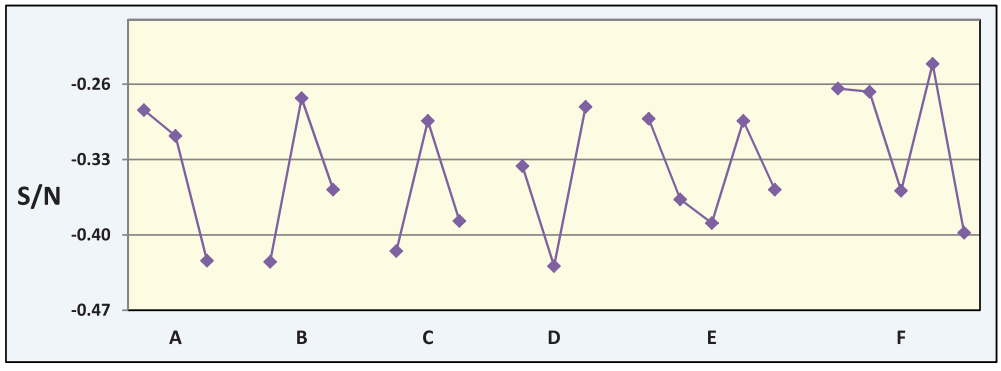

Figure 12. Mean RPD ratio plot for each level of the factors in hybrid GA-VNS.

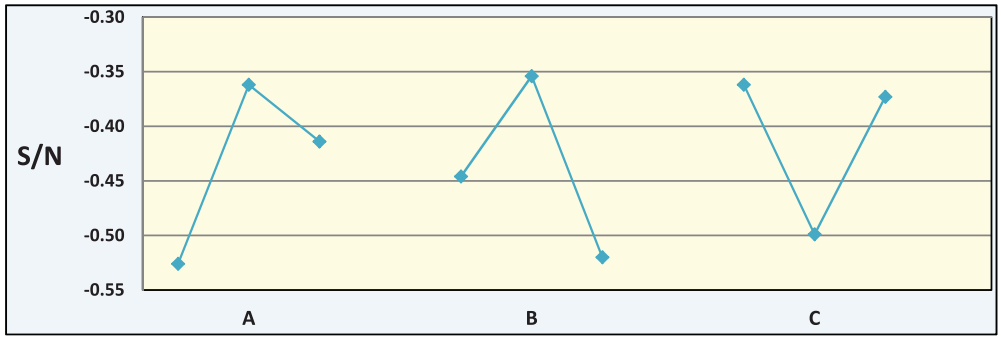

Figure 13. Mean RPD ratio plot for each level of the factors in hybrid VNS-SA.

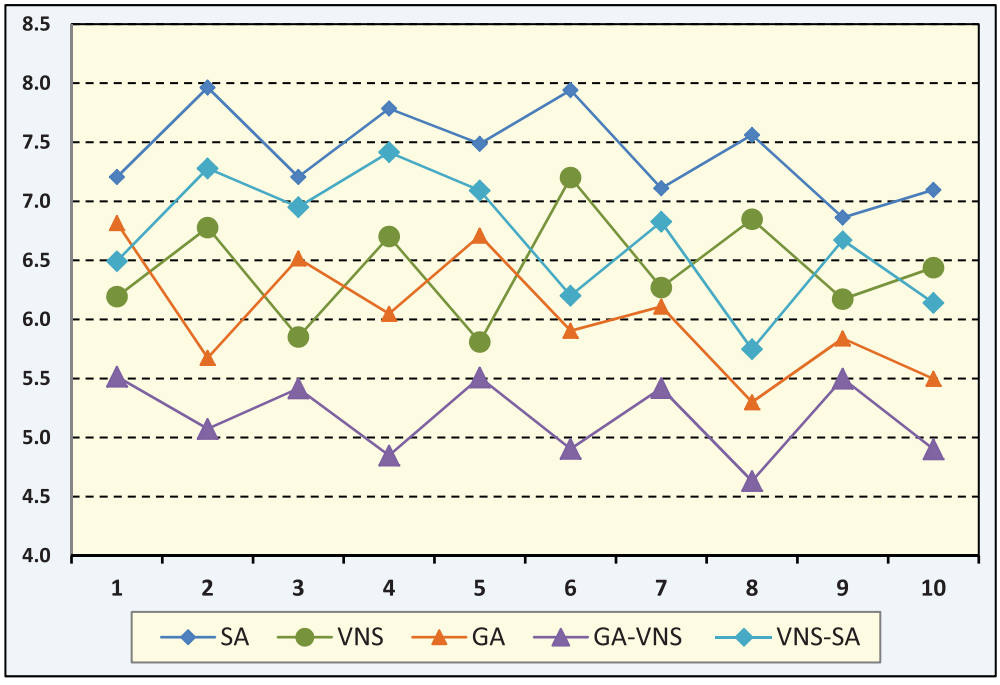

Figure 14. Means plot for the interaction between each algorithm and problem size.

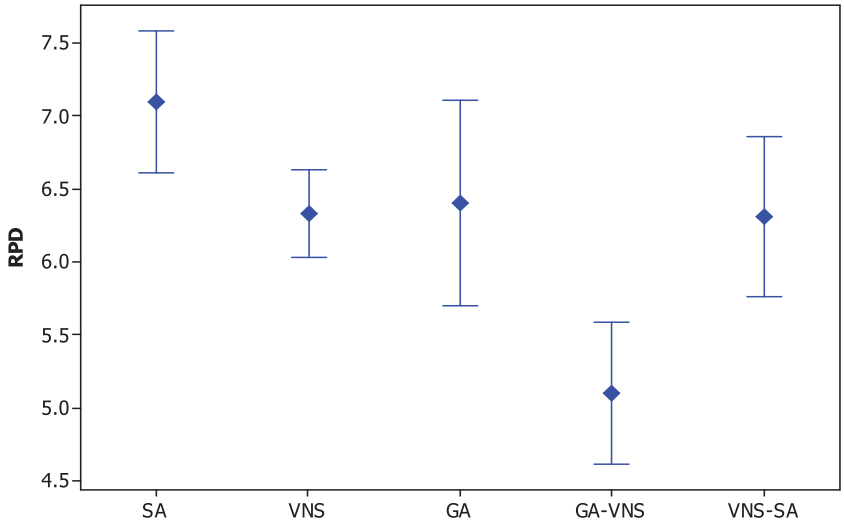

Figure 15. Means plot and LSD intervals for the algorithms.

of the 10 problems considered is illustrated in Figure 14. Each point in the corresponding series described in Figure 14 provides the average obtained from the 200 instances described above per algorithm and problem size. Higher capabilities should lead to consistently lower values for the best performing algorithm. In this regard, note the robust performance exhibited by the GA as the problem size increases. More importantly, this figure illustrates the remarkable performance of the GA-VNS algorithm, particularly when considering larger size problems. Indeed, the GA-VNS algorithm outperforms the VNS-SA one throughout the whole series.

The above results are verified when analysing the variance of the average RPD obtained for each algorithm through the 200 instances and the different problem sizes. Figure 15 plots the means and the least significant difference (LSD) intervals (at a 95% confidence level) obtained for each algorithm when performing an analysis of variance (ANOVA). There is a clear statistically meaningful difference between the performance of GA-VNS and that of all the other algorithms. Indeed, GA-VNS outperforms all the other algorithms, which, at the same time, do not exhibit meaningful differences among themselves in their average RPD performances.

Table 6. CPU completion times (in seconds) for each algorithm.

Problem size $S \times M \times I \times N \times J \times L \times K$	SA	VNS	GA	GA-VNS	VNS-SA
$5 \times 2 \times 3 \times 2 \times 5 \times 2 \times 10$	45	36.2	44.6	241	343
$10 \times 2 \times 5 \times 2 \times 10 \times 2 \times 20$	79.8	60.4	77.8	376.9	480.3
$15 \times 2 \times 8 \times 2 \times 15 \times 2 \times 30$	118	99.8	117.4	515	616
$20 \times 2 \times 10 \times 2 \times 20 \times 3 \times 40$	132.7	114.2	131.7	628	730.1
$25 \times 2 \times 15 \times 3 \times 25 \times 3 \times 45$	190	173	191.1	788	890.2
$30 \times 2 \times 50 \times 3 \times 30 \times 3 \times 50$	297.8	279.6	295.6	890.1	993.5
$35 \times 3 \times 60 \times 3 \times 35 \times 4 \times 60$	340	322.4	335.9	932	1036
$40 \times 3 \times 70 \times 3 \times 45 \times 4 \times 75$	425.8	407	424	1021	1221.9
$45 \times 3 \times 80 \times 4 \times 45 \times 4 \times 80$	459	441.5	458.9	1155.3	1357.1
$50 \times 3 \times 100 \times 5 \times 50 \times 4 \times 100$	528.2	505	524.2	1218	1420

We conclude by providing another performance metric that illustrates the superiority of the GA-VNS algorithm over the VNS-SA one. Table 6 describes the CPU completion times (in seconds) required by the different algorithms when they are run using the best levels of the parameters obtained in the previous section. Note how, despite the obvious increase in the time required by both hybrid algorithms, GA-VNS outperforms VNS-SA for all problem sizes.

5. Conclusion and future research directions

We have studied an MSCN design problem with solid transportation. Two hybrid metaheuristics, GA-VNS and VNS-SA (based on the genetic algorithm, variable neighbourhood search and SA frameworks) have been proposed to solve this *NP*-hard problem. Several parameters have been utilised throughout the solution procedure due to the dependency of the algorithms on the choice of parameters. Moreover, the Taguchi parameter design method has been used to tune the parameters and operators of the algorithms.

The experimental results obtained indicate that the GA-VNS algorithm is robust and superior to the other competing methods considered. There are still many potential ideas that researchers could explore in this field, such as applying the hybrid methods proposed in a fuzzy environment. In addition, these metaheuristics can be applied to multi-objective, multi-stage, multi-product SCN design problems. Furthermore, comparisons with other metaheuristic algorithms and exact methods such as branch and bound, Bender's decomposition and LR would constitute an interesting extension of this research.

Acknowledgments

We would like to express our sincere gratitude to the editors and the anonymous referees for their helpful comments and suggestions.

Disclosure statement

No potential conflict of interest was reported by the authors.

References

Alfares, H. K. (2015). Maximum-profit inventory model with stock-dependent demand, time-dependent holding cost, and all-units quantity discounts. *Mathematical Modelling and Analysis, 20*(6), 715–736.

Altiparmak, F., Gen, M., Lin, L., & Karaoglan, I. (2009). A steady-state genetic algorithm for multi-product supply chain network design. *Computers & Industrial Engineering, 56*(2), 521–537.

Altiparmak, F., Gen, M., Lin, L., & Paksoy, T. (2006). A genetic algorithm approach for multi-objective optimization of supply chain networks. *Computers & Industrial Engineering, 51*, 197–216.

Amiri, A. (2006). Designing a distribution network in a supply chain system: Formulation and efficient solution procedure. *European Journal of Operational Research, 171*(2), 567–576.

Che, Z. H. (2012). Clustering and selecting suppliers based on simulated annealing algorithms. *Computers & Mathematics with Applications, 63*, 228–238.

Costa, A., Celano, G., Fichera, S., & Trovato, E. (2010). A new efficient encoding/decoding procedure for the design of a supply chain network with genetic algorithms. *Computers & Industrial Engineering, 59*, 986–999.

Eckert, C., & Gottlieb, J. (2002). Direct representation and variation operators for the fixed charge transportation problem. In J.J.M. Guervós, P. Adamidis, H.G. Beyer, H.P. Schwefel , & J.L. Fernández-Villacañas, (Eds.) *Proceedings of the 7th International Conference on Parallel Problem Solving from Nature*, Lecture Notes in Computer Science (Vol. 2439, pp. 77–87).

Eskandarpour, M., Hessameddin Zegordi, S., & Nikbakhsh, E. (2013). A parallel variable neighborhood search for the multi-objective sustainable post-sales network design problem. *International Journal of Production Economics, 145*, 117–131.

Gen, M., & Cheng, R. W. (1997). *Genetic algorithms and engineering design* (2nd ed.). New York, NY: Wiley& Sons.

Gen, M., & Cheng, R. W. (2000). *Genetic algorithms and engineering optimization*. New York, NY: John Wiley & Sons.

Gen, M., Altiparmak, F., & Lin, L. (2006). A genetic algorithm for two-stage transportation problem using priority-based encoding. *OR Spectrum, 28*, 337–354.

Hajiaghaei-Keshteli, M. (2011). The allocation of customers to potential distribution centers in supply chain network: GA and AIA approaches. *Applied Soft Computing, 11*, 2069–2078.

Hajiaghaei-Keshteli, M., Molla-Alizadeh-Zavardehi, S., & Tavakkolimoghaddam, R. (2010). Addressing a nonlinear fixed-charge transportation problem using a spanning tree-based genetic algorithm. *Computers & Industrial Engineering, 59*, 259–271.

Jablonsky, J. (2007). Measuring the efficiency of production units by AHP models. *Mathematical and Computer Modelling, 46*(7–8), 1091–1098.

Jang, Y. J., Jang, S. Y., Chang, B. M., & Park, J. (2002). A combined model of network design and production/distribution planning for a supply network. *Computers & Industrial Engineering, 43*, 263–281.

Jayaraman, V., & Pirkul, H. (2001). Planning and coordination of production and distribution facilities for multiple commodities. *European Journal of Operational Research, 133*, 394–408.

Khalifehzadeh, S., Seifbarghy, M., & Naderi, B. (2015). A four-echelon supply chain network design with shortage: Mathematical modeling and solution methods. *Journal of Manufacturing Systems, 35*, 164–175.

Kirkpatrick, S., Gelatt Jr., C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. *Science, 220*, 671–680.

Kovács, G., & Marian, M. (2002). Viability results in control of one-dimensional discrete time dynamical systems defined by a multi-function. *Pure Mathematics and Applications, 13*(1–2), 185–195.

Kristianto, Y., Gunasekaran, A., Helo, P., & Hao, Y. (2014). A model of resilient supply chain network design: A two-stage programming with fuzzy shortest path. *Expert Systems with Applications, 41*, 39–49.

Lančinskas, A., Martínez Ortigosa, P., & Žilinskas, J. (2015). Parallel optimization algorithm for competitive facility location. *Mathematical Modelling and Analysis, 20*(5), 619–640.

Lee, J. E., Gen, M., & Rhee, K. G. (2009). Network model and optimization of reverse logistics by hybrid genetic algorithm. *Computers & Industrial Engineering, 56*, 951–964.

Lin, L., Gen, M., & Wang, X. (2009). Integrated multi-stage logistics network design by using hybrid evolutionary algorithm. *Computers & Industrial Engineering, 56*, 854–873.

Liu, L., Yang, X., Mu, H., & Jiao, Y. (2008). The fuzzy fixed charge transportation problem and genetic algorithm. In J. Ma, Y. Yin, J. Yu, & S. Zhou (Eds.)*Proceedings of the 5th International Conference on Fuzzy Systems and Knowledge Discovery*, 208–212. Jinan Shandong, China :IEEE.

Lotfi, M. M., & Tavakkoli-Moghaddam, R. (2013). A genetic algorithm using priority-based encoding with new operators for fixed charge transportation problems. *Applied Soft Computing, 13*, 2711–2726.

Mehdizadeh, E., Afrabandpei, F., Mohaselafshar, S., & Afshar-Nadja, B. (2013). Design of a multi-stage transportation network in a supply chain system: Formulation and efficient solution procedure. *Scientia Iranica E, 20*(6), 2188–2200.

Melo, M. T., Nickel, S., & Saldanha-da-Gama, F. (2014). An efficient heuristic approach for a multi-period logistics network redesign problem. *TOP, 22*(1), 80–108.

Michalewicz, Z., Vignaux, G. A., & Hobbs, M. (1991). A non-standard genetic algorithm for the nonlinear transportation problem. *ORSA Journal on Computing, 3*, 307–316.

Mladenovic, N., & Hansen, P. (1997). Variable neighborhood search. *Computers & Operations Research, 24*, 1097–1100.

Molla-Alizadeh-Zavardehi, S., Sadi Nezhad, S., Tavakkoli-Moghaddam, R., & Yazdani, M. (2013). Solving a fuzzy fixed charge solid transportation problem by metaheuristics. *Mathematical and Computer Modelling, 57*, 1543–1558.

Olivares-Benitez, E., Gonza´lez-Velarde, J. L., & Rı´os-Mercado, R. Z. (2012). A supply chain design problem with facility location and bi-objective transportation choices. *TOP, 20*, 729–753.

Olivares-Benitez, E., Rı´os-Mercado, R. Z., & Gonza´lez-Velarde, J. L. (2013). A metaheuristic algorithm to solve the selection of transportation channels in supply chain design. *International Journal of Production Economics, 145*, 161–172.

Phadke, M. S. (1989). *Quality engineering using robust design*. Upper Saddle River, NJ: Prentice-Hall.

Sanz-García, A., Pernía-Espinoza, A., Fernández-Martínez, R., & Martínez-de-Pisón-Ascacíbar, F. J. (2012). Combining genetic algorithms and the finite element method to improve steel industrial processes. *Journal of Applied Logic, 10*, 298–308.

Souza dos Santos, D. P., & Silva Formiga, J. K. (2014). Application of a genetic algorithm in orbital maneuvers. *Computational and Applied Mathematics*, 34(2), 437–450. DOI 10.1007/s40314-014-0151-x

Stankiewicz, W., Roszak, R., & Morzyński, M. (2011). Genetic algorithm-based calibration of reduced order Galerkin models. *Mathematical Modelling and Analysis, 16*(2), 233–247.

Syam, S. S. (2002). A model and methodologies for the location problem with logistical components. *Computers & Operations Research, 29*, 1173–1193.

Syarif, A., Yun, Y., & Gen, M. (2002). Study on multi-stage logistic network: A spanning tree-based genetic algorithm approach. *Computers & Industrial Engineering, 43*, 299–314.

Vizvári, B., Lakner, Z., Csizmadia, Z., & Kovács, G. (2011). A stochastic programming and simulation based analysis of the structure of production on the arable land. *Annals of Operations Research, 190*(1), 325–337.

Wu, A., Yu, H., Jin, S., Lin, K., & Schiavone, G. (2004). An incremental genetic algorithm approach to multiprocessor scheduling. *IEEE Transactions on Parallel and Distributed Systems, 15*, 824–834.

Yeh, W. C. (2005). A hybrid heuristic algorithm for the multi-stage supply chain network problem. *International Journal of Advanced Manufacturing Technology, 26*, 675–685.

Yeh, W. C. (2006). An efficient memetic algorithm for the multi-stage supply chain network problem. *International Journal of Advanced Manufacturing Technology, 29*, 803–813.

Zhou, G., Min, H., & Gen, M. (2002). The balanced allocation of customers to multiple distribution centers in the supply chain network: A genetic algorithm approach. *Computers & Industrial Engineering, 43*, 251–261.