A Hybrid metaheuristic algorithm for the optimal design of multi-objective supply chain systems

Problem statement

The schematic representation of a typical supply chain is shown in Fig. 1. It starts with the suppliers who supply the spare parts and components or raw materials to the manufacturing plants. Then, the manufacturing plants transform the raw material into useful products. After that, the products are shipped to warehouses for storage. Then, the products are shipped to distribution centers for delivery to the end customers.

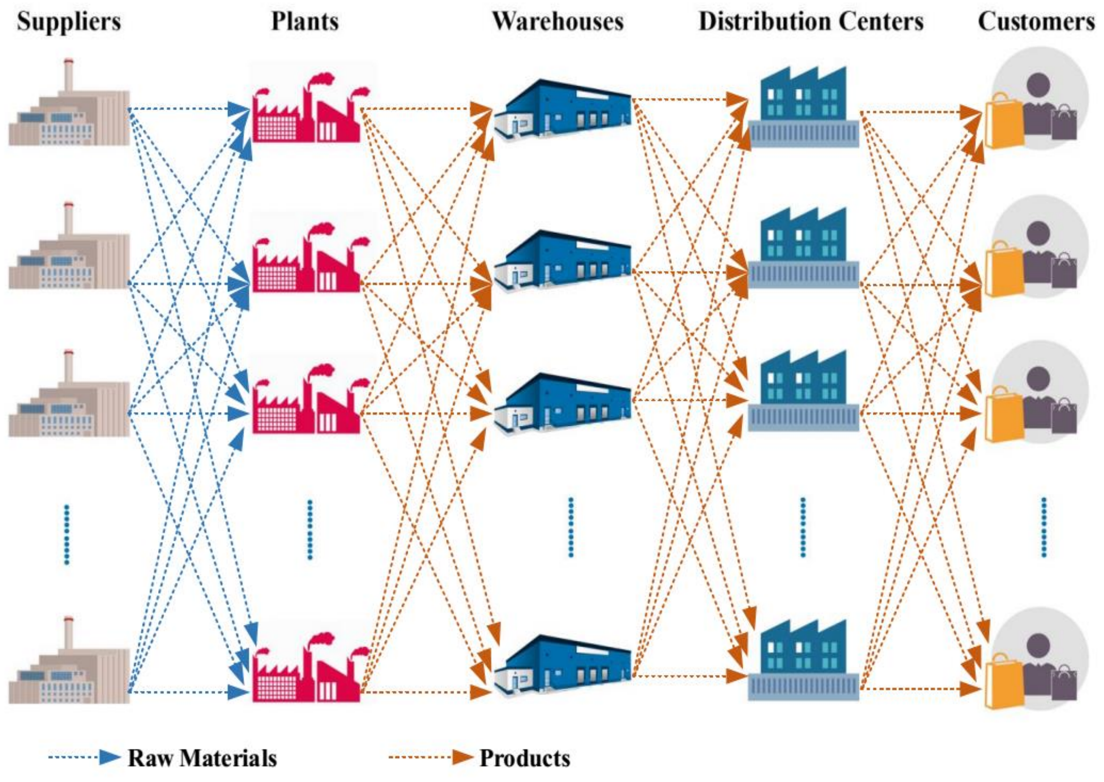

Fig.1. The typical supply chain network

In this paper, it is a requirement to select the optimal number of suppliers and their optimal locations, determine the number and locations of the factories, warehouses, and distribution centers, the flow between the facilities to maximize the total profit of the supply chain, minimize the total supply chain risk, and minimize the total environmental impact that meets the customer's demands and all of the capacities. The assumptions used in this problem are: (1) the time planning horizon is fixed, (2) the capacities and locations of the facilities are known, (3) demands of the customers are fixed and known, (4) the factories, distribution centers, and warehouses have limited capacities, (5) the manufacturing plants, warehouses, and distribution centers operate perfectly. They are not subject to failures as in (Pasandideh, S.H.R. et. al. 2015), and (6) the demand is a fixed amount over the entire price.

Mathematical model

In this section the multi-objective mixed integer linear programming model is presented. The following notations are adopted for the model formulation:

Sets	
S	set of suppliers (1,2…s..S)
I	set of potential factories (1 . . i . I)
J	set of potential warehouses (1,2..j..J)
K	set of potential distribution centers (1,2..k..K)
M	set of demand markets (1 . . . m . . . M)
P	set of products (1 . . p . P)
T	set of raw materials (1 . . t . T)

Parameters	
CP_{ip}	cost of producing product p at plant I;
CC_{st}	purchase cost of raw material t from supplier s;
CB_{sit}	transportation cost of raw material t per km between supplier s and plant i ;
CT_{ijp}	transportation cost of product p per km between plant i and warehouse j;
CD_{jkp}	transportation cost of product p per km between warehouse j and distribution center k;
CO_{kmp}	transportation cost of product p per km between distribution center k and demand market m ;
E_i	fixed cost for opening plant i;
F_j	fixed cost for opening warehouse j;
G_k	fixed cost for opening distribution center k;
Cas_{st}	capacity of supplier s for raw material t;
Cap_{ip}	capacity of plant i for product p;

Caw_{jp}	capacity of warehouse j for product p;
Cad_{kp}	capacity of distribution center k for product p;
t_{si}	the distance between supplier s and plant at location i generated based on the Euclidean distance;
t_{ij}	the distance between plant at location i and warehouse at location j generated based on the Euclidean distance;
t_{jk}	the distance between warehouse at location j and distribution center at location k generated based on the Euclidean distance;
t_{km}	the distance between distribution center at location k and market m generated based on the Euclidean distance;
D_{mp}	demand of customer m for product p;
w_{tp}	yields of product p from raw material t;
EF_{i}	environmental impact of opening plant at location i;
EW_{j}	environmental impact of opening warehouse center at location j;
ED_{k}	environmental impact of opening distribution center at location k;
EP_{i}	environmental impact caused by production at plant i;
ETS_{sit}	environmental impact per unit and per distance caused by transporting raw material t from supplier s to plant i;
ETP_{ijp}	environmental impact per unit and per distance caused by transporting product p from plant i to warehouse j;
ETD_{jkp}	environmental impact per unit and per distance caused by transporting product p from warehouse j to distribution center k;
ETW_{kmp}	environmental impact per unit and per distance caused by transporting product p from distribution center k to market m;
sp_{kmp}	selling price of product p transported from distribution center k at market m;
Prd_{st}	probability of delivery risk for raw material t from supplier s;
Prq_{st}	probability of quality risk for raw material t from supplier s;
Prd_{ip}	probability of delivery risk for product p from plant i;
Prq_{ip}	probability of quality risk for product p produced at plant i;
Prd_{jp}	probability of delivery risk for product p from warehouse j;
Prd_{kp}	probability of delivery risk for product p from distribution center k;
IRD_{st}	impact caused by risk of delivery for raw material t from supplier s;
IRQ_{st}	impact caused by risk of quality for raw material t from supplier s;

IRD_{ip}	impact caused by risk of delivery for product p from plant i;
IRQ_{ip}	impact caused by risk of poor quality for product p from plant i;
IRD_{jp}	impact caused by risk of delivery for product p from warehouse j;
IRD_{kp}	impact caused by risk of delivery for product p from distribution center k.
Decision variables	
X_{sit}	quantity of raw material t supplied by supplier s to plant i
Y_{ijp}	quantity of product p produced at plant i shipped to warehouse centre j
Z_{jkp}	quantity of product p transported from warehouse j to distribution center k
Q_{kmp}	quantity of product p transported from distribution center k to market m
X_i	1, if a plant is located and set up at potential site i, 0, otherwise
Y_j	1, if a warehouse is located and set up at potential site j, 0, otherwise
Z_k	1, if a distribution center is located and set up at potential site k, 0, otherwise
U_s	1, if a supplier s is selected, 0, otherwise

In this model, three objectives are considered. The first objective (Z_1) is profit maximization. The second objective (Z_2) is the minimization of risk and the third objective (Z_3) is the minimization of the supply chain emissions.

$$\text{Max } Z_1 = \sum_k \sum_m \sum_p sp_{kmp} Q_{kmp} - (\sum_i Ei\, X_i + \sum_j F_j Yj + \sum_k Gk\, Z_k + \sum_s \sum_t \sum_i (CC_{st} + CB_{sit} t_{si}) X_{sit} + \sum_i \sum_j \sum_p \left(CP_{ip} + CT_{ijp} t_{ij}\right) Y_{ijp} + \sum_j \sum_k \sum_p CD_{jkp} t_{jk} Z_{jkp} + \sum_k \sum_m \sum_p CO_{kmp} t_{km} Q_{kmp}) \quad (1)$$

$$\text{Min } Z_2 = \sum_s \sum_t (Prd_{st} IRD_{st} + Prq_{st} IRQ_{st} + Prd_{st} Prq_{st} \text{Max}(IRD_{st}, IRQ_{st})) U_s + \sum_i \sum_p (Prd_{ip} IRD_{ip} + Prq_{ip} IRQ_{ip} + Prd_{ip} Prq_{ip} \text{Max}\left(IRD_{ip}, IRQ_{ip}\right)) X_i + \sum_j \sum_p Prd_{jp} IRD_{jp} Y_j + \sum_k \sum_p Prd_{kp} IRD_{kp} Z_k \quad (2)$$

$$\text{Min } Z_3 = \sum_i EFi\, X_i + \sum_j EW_j Yj + \sum_k EDk\, Z_k + \sum_i \sum_j \sum_p EP_{ip} Y_{ijp} + \sum_s \sum_i \sum_t ETS_{sit}\, t_{si} X_{sit} + \sum_i \sum_j \sum_p ETP_{ijp} t_{ij} Y_{ijp} + \sum_j \sum_k \sum_p ETD_{jkp} t_{jk} Z_{jkp} + \sum_k \sum_m \sum_p ETW_{kmp} t_{km} Q_{kmp} \quad (3)$$

Subjected to

$$\sum_k Q_{kmp} = D_{mp} \quad \forall\, m,p \quad (4)$$

$$\sum_{i} X_{sit} \leq U_s Cas_{st} \quad \forall\, s,t \tag{5}$$

$$\sum_{s}\sum_{t} w_{tp} X_{sit} \leq X_i Cap_{ip} \quad \forall\, i,p \tag{6}$$

$$\sum_{i}\sum_{p} Y_{ijp} \leq Y_j \sum_{p} Caw_{jp} \quad \forall\, j \tag{7}$$

$$\sum_{j}\sum_{p} Z_{jkp} \leq Z_k \sum_{p} Cad_{kp} \quad \forall\, k \tag{8}$$

$$\sum_{s}\sum_{t} w_{tp} X_{sit} = \sum_{j} Y_{ijp} \quad \forall\, i,p \tag{9}$$

$$\sum_{i} Y_{ijp} = \sum_{k} Z_{jkp} \quad \forall\, j,p \tag{10}$$

$$\sum_{j} Z_{jkp} = \sum_{m} Q_{kmp} \quad \forall\, k,p \tag{11}$$

$$U_s, X_i, Y_j\ , Z_k \in \{0, 1\} \quad \forall s, i,j,k \tag{12}$$

$$X_{sit}, Y_{ijp}, Z_{jkp}, Q_{kmp} \geq 0 \quad \forall\, i, j,k, m,p,t \tag{13}$$

The first objective tries to maximize the total profit. The first term of Eq. (1) represents the total revenue. The second, third, and fourth terms represent the fixed cost of establishing facilities in the plant, warehouse centers, and DCs, respectively. The fifth term is the purchase cost and transportation cost of shipped quantities from suppliers to plants. The sixth term is the production cost at plant and the transportation cost from plants to warehouse centers. The last two terms represent the transportation cost from warehouse to DC and from DC to market, respectively.

The second objective (Z_2) is to minimize the supply chain risk. The first term represents the expected quality and delivery risks caused by supply the raw material from supplier. The second term is the expected quality and delivery risks caused by producing the product at plaint. The third and fourth terms represent the delivery risk caused by shipped the products from warehouse and DC centers, respectively.

The third objective (Z_3) is to minimize the total supply chain emissions. The first three terms represent the emissions caused by the establishing of plants, warehouses, and distribution centers facilities, respectively. The fourth term represents the emissions caused by production of products at plants. The other terms are the emissions due to the transportation.

Equation (4) represents the demand constraint. Equations (5-8) represent the capacity constraints of the suppliers, plants, warehouse centers, and distribution centers, respectively. Equations (9-11) represent the material balance constraints. Equation (12) represents the nature of the binary variables. Equation (13) defines the values of the continuous variables.

The aim of this research is to propose a new hybrid metaheuristic algorithm based on a combination of tabu search, simulated annealing, and neighborhood search algorithms for the optimal design of a common problem, which is a multi-objective supply chain system. The literature review indicates that there is still a lack of development of an efficient algorithm for optimizing a multi-objective supply chain network. Specifically, there has been no work test and combine the strengths of three algorithms in a hybrid algorithm in supply chain field.
In research, firstly we explore the characteristics of tabu search algorithm, simulated annealing, and neighborhood search algorithm. The advantages of these solution algorithms are used to develop the new hybrid algorithm. Then, the developed algorithm will be used to design multi-objective supply chain model. The details of the characteristics of the three algorithms are explored and provided in subsequent sections.

Tabu search algorithm

TSA is an iterative process developed by Glover [5]. Information and knowledge about the selective solutions during the search are maintained by flexible memory structures which is the tabu search core. Based on the search history and the current solution properties, a search direction is determined in the feasible space. Tabu search keeps track of the moves made before arriving at the current solution so that cycling will not occur as the algorithm proceeds. This is what distinguishes tabu search from other search approaches. The tabu solutions are recorded on a list referred to as the 'tabu list' and remain on the list for a short period. The tabu search algorithm pseudo-code is given in algorithm 1.

Pseudocode [1]

Algorithm 1. Tabu Search algorithm

```
1:  Generate an initial solution ($S_0$);
2:  Initialize the algorithm parameters; maximum iterations number,
    tabu size, and neighbors number;
3:  While stopping criterion is not satisfied Do
4:          Generate a set of neighborhood solutions $N(S_0)$ for the
            current solution via neighborhood structure
5:          Evaluate each neighbor of the current solution
6:          Select m best non-tabu solutions neighbors among the
            explored solutions
7:          If the explored neighborhood is empty, Then
8:              Backtracking is applied
9:              If no solution is identified by backtracking, Then
10:             The Algorithm is restarted
11:         Update the current solution; update the tabu list; keep the
            best  h-1 solutions
12:         Repeat the previous steps until the stopping criterion is
            satisfied
13: End while
14: Report final non-dominated frontier
```

Advantages

- The merits inherited in TS to explore the global optima. Aspiration and diversification play an important role in TS. To improve accuracy, an aspiration criterion is considered to ignore the Tabu condition of the selected move. Even though the selected best is prohibited by the Tabu condition, the condition is overridden when the selected move results in asolution better than any solutions visited so far. Tabu list and aspiration enables a TS to find the global optimal solution with good probability.
- It allows search to override the tabu status of the solution and also provide backtracking of recent solutions as they lead to a new path towards a better solution. Diversification is generally used to explore subdomains that may not be reached otherwise. Thus, the search is redirected from a different initial solution

Disadvantages

- The main drawback associated with tabu search is cycling, in which it follows the same path unless a tabu neighbor exists. Alternatively, it can be said that a loop will be encountered if the search moves to a previously visited solution that has not been tabu for the last two iterations [2].
- Random points are selected from neighborhoods around the current solution and no local search is performed in the random variable neighborhood search [3].

Simulated annealing algorithm

The first use of SA for solving combinatorial problems was by Kirkpatrick et al. [4]. In the SA algorithm, the material is heated and then is cooled slowly. The material grain sizes are affected by cooling rate. Through this mechanism, changes in energy are simulated until a frozen steady state is reached. In this paper, the behavior of SA is exploited to obtain a high-quality solution in the term of convergence. The transition from one solution to another solution is stated by changing a single facility status in a stochastic mechanism (e.g. 1 and 0 represent open and close facility status, respectively). The facility is chosen at random. If the generated solution is non-dominated compared with the current solution, the current solution is replaced by the new solution. Otherwise, we can accept the generated solution with the following probability:

$$P(A) = exp\ ((-\Delta E) / T) \qquad (1)$$

where $$\Delta E = [(Z - Z) / Z] \times 100, \qquad (2)$$

T is the temperature, Z' is the weighted objective functions value of the generated solution, and Z is weighted objective functions value of the current solution. Within the number of iterations (K), the temperature is fixed. Then, it is reduced by a cooling rate (α), where $\alpha \in (0,1)$. The simulated annealing pseudo-code is given in algorithm 2.

Pseudocode [6]

Algorithm 2 Simulated Annealing algorithm

```
1: Generate an initial solution ($S_0$)
2: Initialize the algorithm parameters T, α, K, and x
   // where T is the initial temperature //
   // α is the cooling rate //
   // K is the number of iterations at a fixed temperature //
   // x is a uniform random number ∈ (0,1) //
3: **While** stopping condition is not satisfied **do**
4:     **For** i ← 1: K
5:         Generate a new neighbor for the current solution
           via neighborhood structure  and evaluate it
6:         Compare the new neighbor against the current
           solution
7:         **If**   the neighbor is non-dominated, **Then**
8:             Add the neighbor to the non-dominated frontier
9:             Replace the current solution with the neighbor
10:        **Else**   check metropolis criterion
11:            **If** $x < P(A) = \exp^{((-\Delta E)/T)}$ , **Then**
12:                Replace the current solution with the neighbor
13:            **End if**
14:        **End if**
15:    **Next i**
16:    T ← T×α
17: **End while**
18: **Return** the final non-dominated frontier
```

Advantages

- Simulated annealing and tabu search have both strong local search ability and global search ability.
- The main features of SA that make this algorithm more sophisticated are perturbation annealing schedule and transition probability. Perturbation generates a new solution,

annealing schedule control the initial temperature, final temperature and rate of cooling while transition probability help heuristic to escape local optima

Disadvantages

- It is still very difficult to select an appropriate cooling scheme to balance computational time and final solution quality. If annealing occurs too slowly, it may result in long computational time. If annealing occurs too quickly, the search can miss solution spaces that may contain-high quality solutions [3].
- Random points are selected from neighborhoods around the current solution and no local search is performed in the random variable neighborhood search
- It requires large number of iterations to generate an optimal or near optimal solution. In addition, SA has no concept of short-term memory list of prohibited neighboring solutions as in tabu search algorithm and hence the possibility of revisiting the solution increased. These two drawbacks posed by SA leads to more number of iteration and thus longer computational time to generate the global optima solution [7].

Neighborhood search algorithm

Variable Neighborhood Search (VNS) is a metaheuristic framework that was introduced by Mladenović and Hansen [8] for solving (combinatorial) optimization problems. Given an incumbent solution *x*, the strategy of VNS consists in systematically visiting increasingly distant *k*-th neighborhoods of the current incumbent solution *x*, i.e., $N_k(x)$. The new neighborhoods are then explored by a local search method in order to identify the local optima. A new solution is accepted if and only if an improvement is made.

VNS starts with a feasible initial solution. Then, it is manipulated through a multi-nested loop by two main functions: shaking (SN) and local search (LS). Both functions attempt to improve the solution by exploring neighborhood structures. The neighborhood structure is used to move from the current solution to one of its neighbors to generate a new solution. The variable neighborhood search algorithm pseudo-code is given in algorithm 3.

Pseudocode [9]

Algorithm 3. Variable neighborhood search algorithm

```
Generate an initial solution
Set k ( neighborhood structure)=1, s ( number of iterations)= 0
```

```
While s < S do:
    Set k = 1
  While $k \le k_0$ do:
      Shaking: use neighborhood structure $SN_k$ on the current solution y
      to randomly generate a solution y′.
        Local search:
          For j = 1 to $j_0$:
            Neighborhood exploration: use neighborhood structure $LS_j$ on
            the current point y′ to randomly generate a neighbor y″.
            Move to y″ only if it gives a better solution:
              If $f(y'') < f(y')$, set $y' \leftarrow y''$.
          End for
      Solution change: change the current solution y to the new solution y′
      subject to the following condition:
        If $f(y') < f(y)$, set $y \leftarrow y'$, k = 1, and s = 0.
        Otherwise    set k = k + 1, If $k > k_0$ set s = s + 1.
  End while
End while
Output. The final solution y is the best one, i.e., the current of the last iteration.
```

Advantages

- Random points are selected from neighborhoods around the current solution and local search is performed in the random variable neighborhood search to explore local solutions

Disadvantages

- It has no concept of short-term memory list of prohibited neighboring solutions as in tabu search algorithm and hence the possibility of revisiting the solution increased (Cycling)
- The global solution, as provided by the Tabu search and simulated annealing algorithms, cannot be guaranteed.

The hybrid TSA-SA-NSA algorithm characteristics

The aim of this study is to hybridize three meta-heuristic methods: NSA, SA, and TS. Our goal is to aggregate the approaches in order to benefit from the advantages of these solution algorithms. The proposed TSA-SA-NSA meta-heuristic is developed for the design of multi-objective supply chain network problems. The proposed TSA-SA-NSA has several advantages, including a stochastic feature that avoids cycling and a tabu list to escape from local optima, as well as a random variable neighborhood search to investigate local solutions. These characteristics significantly improve the performance of the traditional SA, TSA, and NSA by limiting the search to a previously visited solution. The hybrid algorithm begins at the initial temperature and continues until the ultimate temperature is reached. By multiplying the temperature by a cooling rate less than one, the temperature drops at an exponential rate. At each temperature, the proposed algorithm performs a number of iterations, and new solutions are accepted if they dominate the current solution or are accepted with probability P at each iteration. The tabu search algorithm is in charge of coming up with a new solution based on the existing one. The neighborhood structure, which is modified by two major functions through a multi-nested loop, is used to move from one solution to the next. Shaking and local search are two of these features. Both functions use neighborhood structures to improve the solution. Tabu search keeps track of the steps taken before arriving at the current solution, ensuring that the algorithm does not cycle. The characteristics of the proposed algorithm are explained in below:

1. **Neighborhood generation**

To construct a new solution, the neighborhood structure attempts to transform the existing solution into one of its neighbors. To generate a new solution, the developed algorithm employs two neighborhood structure phases. Shaking is the initial phase, which generates a number of pre-selected neighborhood structures (moves) around the existing solution. In the intensification step, a local search is used to improve the provided solution. Many neighborhood structures will be examined in our algorithm, and the best structures will be selected.

2. **Transition probability**

The fitness value of the neighboring solution that will be generated at random is assessed. The new solution is acceptable if it is non-dominated when compared to the current solution during the fitness function evaluation; otherwise, we can accept the new solution with the following probability:

$$P(A) = exp\ ((-\Delta E) / T)$$

where $\Delta E = [(Z - z) / z'] \times 100,$

T is the temperature, Z' is the weighted objective functions value of the generated solution, and Z is weighted objective functions value of the current solution.

3. **Annealing schedule**

The cooling schedule determines the annealing temperature and transition probability for each iteration. The quality of the solution is also affected by cooling. The superior solution is accepted if the cooling rate is lower. In such instances, however, the convergence rate will be modest. The following cooling schedule is followed:

$T(K) = T \times \alpha$

Within the number of iterations (K), the temperature is fixed. Then, it is reduced by a cooling rate (α), where $\alpha \in (0,1)$.

4. **Tabu list**

A tabu list is used to keep track of solutions that have been visited. The tabu list restricts the search from returning to the selected points before reaching an expiration point n, which is also equal to the length of a move's tabu tenure. The list's length is fixed. To improve the current solution, a tabu move may be revoked. This is accomplished by using an aspiration criterion.

5. **Backtracking mechanism**

The search may become stuck at a local minimum during an iteration of the proposed algorithm. The reason for this is that the explored neighborhood's set is now empty. When there are no other options or the tabu movements cannot be revoked, this happens. When a neighborhood has been explored, a number of best neighbors (e.g. h) of the solution are preserved to solve this problem. To reduce computational complexity, we set h to three in this study. We examine the previously selected solutions regardless of the fitness function value in the backtracking approach. If all of the solutions S1,...,Sj-1 and $1 < j \leq h$ fail, the search is resumed from the jth neighbor. In the worst-case scenario of using the backtracking approach, the search remains at the local minimum. In this case, we restart the proposed algorithm with a new starting solution by changing the status of the facilities in the previous one (S_0). The algorithm is only restarted once over the entire search procedure.

6. **Aspiration** (which property will be added, aspiration or backtracking or both)

This Tabu search variable prevents the search from becoming stuck in a solution surrounded by Tabu neighbors. If the search becomes trapped in a solution surrounded by just Tabu solutions at any point, the solution with an objective function better than aspiration is chosen for further investigation.

7. Stopping criteria

The TS,SA,NSA method will use the following stopping criteria to prevent the searching mechanism from roaming into the solution space:

- Number of iterations: Any further drop in temperature would be ineffective since the chance of accepting a suboptimal solution is very minimal at low temperatures, and the outcomes are nearly indistinguishable from the optimal solution.
- Variable reject: When the reject counter reaches a fixed value, it signifies that no optimal or near-optimal solution has been explored in the previous steps. As a result, the chances of finding a better solution are low, and the search process is stopped.

Pseudocode of the proposed algorithm

Algorithm 4. **SA-TA- NSA**

```
Generate an initial solution ($S_0$)
Initialize the algorithm parameters T, $T_f$ , α, K, SNj , LSi,  tabu size, number of neighbors, and $x$
Do While T > $T_f$
    Do while k ≤ K
      For j = 1 : $J_0$
        Shaking: Generate a set of neighborhood solutions N($S_0$) for the current solution using neighborhood
                    structure  SNj
         Evaluate each neighbor of the current solution
         Select h best non-tabu solutions neighbors among the explored solutions
           If the explored neighborhood is empty, then
                  Backtracking is applied
                            If no solution is identified by backtracking, Then
                                The Algorithm is restarted
           If new best solution dominates the current solution
```

```
            the new best solution =  the current solution; update the tabu list; keep the
            best  h-1 solutions
        Else
            If x < P(new best solution) = exp $^{((-\Delta E)/T)}$ , Then
                Replace the current solution with the new best solution
                update the tabu list; keep the best  h-1 solutions
            End If
        End If
        Local search:
            For i = 1 to $I_0$:
                Neighborhood exploration: use neighborhood structure LSi on
                new best solution to randomly generate a neighbor solution.
                Move to neighbor solution only if it dominates the new best solution
                If neighbor solution dominates the new best solution
                    the new best solution =  the neighbor solution; update the tabu list;
                Else
                    If x < P(the neighbor solution) = exp $^{((-\Delta E)/T)}$ , Then
                    Replace the new best solution with the neighbor solution
                    update the tabu list; keep the best  h-1 solutions
                End If
            End for
        Solution change: change the current solution to the new best solution
        subject to the following condition:
        If the new best solution dominates the current solution, set current solution ← new best solution
        , j = 1, k= k +1
        Otherwise  set j = j + 1, If j > $J_0$ set k = k + 1.
        End If
    End for
  End Do while
  T ← T×α
End Do while
```

Return the final non-dominated frontier

Multi-objective technique

In multi-objectives optimization there are more than one objective function to optimize. In our case in this paper we have three competing objectives. In such situation usually there is a trade-off among these objectives, making it difficult to find a single optimal solution that improves all values of the objective functions at the same time. This trade-off frequently results in a set of optimal solutions at the end of the optimization process known as Pareto optimal solutions. In our effort to obtain Pareto optimal solutions, use the random-weight approach proposed byMurata et al., 1996, among others) is utilized for identifying the acceptable solutions to determine the set of the Pareto optimal solutions. This approach is also utilized in (N. Kartina Puji et al., 2017; A. Saman et al., 2013; Mohammed & Duffuaa 2020;Yang et al. 2015). The advantage of this approach is that it takes less computational time than epsilon constraint method. Epsilon constraint method takes long time due to increasing the number of constraints. Another advantage of random weighted sum is that it avoids the local optimal and instead it explores the whole solution space. This leads to giving an equal chance to search all possible Pareto-optimal solutions along the Pareto frontier. The fitness function of the model is defined by combining given m objective functions into a single function as follows:

$$Z = \sum_{i=1}^{m} w_i \, f_i'(x), \text{ i=1, …, m}$$

If R_i are positive numbers selected randomly, the random weights w_i are computed by the following equation:

$$w_i = \frac{R_i}{\sum_{i=1}^{m} R_i}, \text{ i=1,…,m}$$

Where $w_i \geq 0$, i $= 1,2, \dots,$ m and $\sum_{i=1}^{m} w_i = 1$

The objective functions must be normalized as follow:

$$f_i'(\text{x}) = \frac{f_i - f_{min}}{f_{max} - f_{min}} \text{ , } i = 1,2,3$$

Where $f_{max} and\ f_{min}$ are the maximum and minimum values of objective i that obtained by the algorithm as a single objective, respectively.

Solution approach

The problem of designing a supply chain network is a large combinatorial optimization problem, such problems are hard to solve, and they are NP-hard. In this research a novel approach based on tabu search, simulated annealing, and neighborhood search algorithms is developed. The approach for developing the algorithm divides the supply chain network design problem in two phases. The first phase is called the strategic phase in which long term decisions are made such as the selection of the best suppliers and establishing the facilities of the supply chain. These decisions are made once per period and are made by fixing the values of the binary variables. The second phase is concerned with the operational decisions such as quantities of raw material purchased from suppliers, products to be manufactured in plants and amounts to be shipped from suppliers to plants and among facilities.

The proposed meta-heuristic algorithm designs the strategic network in the first phase, and the operational problem is tackled by a linear programming method in the second phase. The benefit of this approach is that it divides the problem into two stages, which reduces the problem space and solution time. The second phase algorithm is run to optimality for each solution in the first phase. In this regard, the optimal operational solution will be identified for each design in the first phase. The integration of the outcomes from both phases yields the overall objective and complete supply network chain network design.

Construction of an initial solution

Then initial solution may reduce the computation time of meta-heuristic algorithms and affect the quality of the solutions obtained by the proposed algorithm. In this regard, we develop an efficient algorithm to construct an initial solution. To obtain a feasible initial solution, two stages are used. The first step is to decide which of the potential facilities should be opened, whereas the quantities of products flow among the opened facilities are determined in the second stage. To determine which facility in each echelon should be open, the facilities are arranged in an increasing order based on the ratio of $\frac{f_i}{b_i} + \sum_j C_{ij}$, where $f_i\ is$ the fixed establishing cost of facility i, b_i is the capacity of facility *i*, and C_{ij} is the cost of transporting the products from facility *i* to j. The facilities are opened in the number necessary to fulfill the total demand, beginning with the facility with the lowest ratio. The distribution flows among facilities are calculated using a linear programming approach on the basis of a set of determined open facilities. Algorithm 5 depicts the pseudo-code for obtaining the first solution.

Algorithm 5. Greedy algorithm

```
Initialize the parameters of algorithm; number of entities
at each stage $i= 1, 2, ...I$, number of echelons $n$, $f_i, b_i, C_{ij}$,
$k = n-1$
 While K≠ 0 do
     Set all echelon k facilities to zeroes
     Arrange the facilities in ascending order according
         to  $P_i = \frac{f_i}{b_i} + \sum_j C_{ij}$
     If total capacity of establishing facilities in
       $k$ less than total demand of $k+1$
      Move the status of facility i to be open
     Else
         $K=k - 1$
End while
Report an initial solution of the binary variables
Based on the binary variable solution, the flows among
open facilities are determined by a linear programming
approach
```

2.2.5 Setting the proposed algorithm parameters

Proper tuning of the proposed algorithm parameters yields high performance under a variety of conditions. As a result, more accurate and consistent solutions are produced. In this regard, the algorithm parameters values are first set to evaluate its performance. There are several parameters that must be set: the size of the tabu list (Tabu size), the number of iterations to perform the algorithm at each temperature, type of shaking structure, type of local search structure, and the proportion of solutions assessed in the neighborhood (Number of neighbors).

The developed algorithm seeks the following characteristics: diversity of solutions, convergence of solutions, and running time. To do this, the algorithm parameters are set using four common performance evaluation measures:

1. Mean ideal distance (MID): to indicate the closeness of the solutions to the ideal solution. It is computed as follows:

   $$MID = \frac{\sum_i c_i}{n}$$

   where $c_i = \| \vec{f} - \vec{f}_{ideal} \|$, n denotes the effective solutions number.
2. Spread of non-dominance solution (SNS): it assesses the solutions variations from ideal solution. It is calculated using the following formula:

   $$SNS = \sqrt{\frac{\sum_{i=1}^{n}(MID - c_i)^2}{n-1}}$$

   SNS with a higher value is preferable.
3. The diversification metric (DM) assesses the range of Pareto optimal solutions. It's calculated by the following formula:

   $$DM = \sqrt{(maxf_{1i} - minf_{1i})^2 + (maxf_{2i} - minf_{2i})^2 + (maxf_{3i} - minf_{3i})^2}$$

   It is preferable to have a higher DM value.
4. Running time

The parameters of the algorithm in this research will be set based on a middle instance for each group. An artificial neural network will be used to describe the relationship between input variables and quality characteristics or output. Then a mathematical programming model will be construct to find the optimal values of input variable.