ZIBELINE INTERNATIONAL™ PUBLISHING

SSN: 2521-0904 (Print)
ISSN: 2521-0440 (Online)
CODEN: EHJNA9

Engineering Heritage Journal (GWK)

DOI: http://doi.org/10.26480/gwk.01.2021.01.11

REVIEW ARTICLE

A HYBRID ALGORITHM TO SOLVE THE FIXED CHARGE SOLID LOCATION AND TRANSPORTATION PROBLEM

Gbeminiyi John Oyewole*, Olufemi Adetunji

Department of Industrial and Systems Engineering, University of Pretoria, Pretoria 0002, South Africa.
****Corresponding Author Email: johnigbem55@yahoo.com***

ARTICLE DETAILS

Article History:

Received 14 January 2021
Accepted 19 February 2021
Available online 16 March 2021

ABSTRACT

In this paper, we propose a Hybrid Algorithm (HA) to solve the Fixed Charge Solid Location and Transportation problem (FCSLTP). The FCSLTP considers the cost of facility location and route fixed costs during transportation planning or load consolidation. The HA integrates two heuristics into the Genetic Algorithm framework to solve the FCSLTP. Genetic operations are used to select the best combination of facility locations while a greedy heuristic which uses some cost relaxations are used for the initial load allocation. An improvement heuristic, a modified stepping stone method, is then used to consolidate load allocations to realize further possible cost savings. Parameters used for the genetic operations were decided through preliminary studies. Computational studies using randomly generated data were performed to compare the HA solutions with the solutions obtained using CPLEX, a commercial solver. Performance comparison was done based on the quality of solution and computing time. The results suggest the solution approach is competitive.

KEYWORDS

Genetic algorithm, Heuristics, Fixed charge solid transportation, Facility location.

1. INTRODUCTION

The basic multi-item distribution or Solid Transportation problem (STP) introduced by and solved by extends the classical transportation problem (Schell, 1955; Haley, 1962). The STP occurs both in manufacturing and the logistics industry. For example, STP is solved in the logistics industry when decisions are to be made on the quantity of products to move from facility locations to depots or warehouses to customer locations given a limited number of transport resources. In addition, STP finds its use in process industries where raw materials from different sources are required to be shipped to particular destinations in order to meet a target demand requirement (Kundu et al., 2017). The basic STP model has been extended to capture some other real-world problems. Some of these problems are encountered during shipping and they include modelling of fixed charges, incremental discounts, price breaks and uncertainties.

As a result, new problem variants are created. Some of these variants are described (Yang and Liu, 2007; Ojha et al., 2010; Halder et al., 2017). A variant of the STP we are interested in is the Fixed Charge Solid Transportation Problem (FCSTP). The FCSTP as noted is concerned with determining the quantity of products to ship from a fixed set of sources to certain destinations using different conveyances while also considering an associated route fixed charge (Sanei et al., 2017). They further stated that since similar problems such as the Fixed Charge Transportation Problem (FCTP) and Step-Fixed Charge Transportation Problem (SFCTP) have been established to be NP-hard problems, the FCSTP which is an extension of the FCTP implicitly becomes more difficult to solve due to the additional conveyances constraints. Recently, a group researchers discussed a solid transportation and location problem which integrates facility location problem with transport conveyances (Das et al., 2019).

However they did not consider the reality of fixed charges in their models. It is well established in the literature that exact solution approaches such as branch and bound or branch and cut, can provide optimal solutions to NP-hard problems of the class of FCSTP. However, their solution may become ineffective as problem size increases. This has encouraged the development of both heuristics and metaheuristics to solve problem variants of the STP. A group researchers developed a genetic algorithm to solve a discounted fixed charge solid transportation problem (Ojha et al., 2010). Some researchers proposed a hybrid metaheuristic which uses Tabu search to solve FCSTP with uncertainties in the problem parameters (Zhang et al., 2016). Some researchers developed heuristics to solve an FCSTP with fuzzy parameters (Chen et al., 2017; Halder et al., 2017). The use of Lagrange relaxation heuristics was applied to solve FCSTP in which there are more than one fixed charges associated with the transportation routes (Sanei et al., 2017). Metaheuristics are currently being utilized to solve NP-hard problem similar to the FCSTP as indicated earlier.

As noted by some researcher metaheuristics, unlike classical heuristics, possess abilities to prevent optimization solutions from being stuck in local optima (Genove and Gulias, 2011; Fernandes et al., 2014). In addition, the multidimensional search patterns of metaheuristics make navigation

Quick Response Code	Access this article online	
	Website: www.enggheritage.com	**DOI:** 10.26480/gwk.01.2021.01.11

Cite the Article: Gbeminiyi John Oyewole, Olufemi Adetunji (2021). A Hybrid Algorithm to Solve the Fixed Charge Solid Location and Transportation Problem. *Engineering Heritage Journal, 5(1): 01-11.*

towards optimal or near-optimal solutions feasible. Metaheuristics, such as the Genetic Algorithm (GA), have been utilized either in their pure or hybrid forms with other improvement heuristics to solve various combinatorial problems such as those in the class of FCTP and FCSTP. Some works such as have shown how pure GA and hybrid GA have been applied to solve fixed charge problem variants (Perez-Salazar et al., 2015; Calvete et al., 2016; Guo et al., 2017; Hiassat et al., 2017; Yousefi et al., 2018; Ghassemi and Hashemi, 2018). A group reserchers extended their previous GA model to solve an FCTP with the introduction of truckload constraints (Balaji et al., 2009). In this paper, we consider a variant of the FCSTP, referred to as the Fixed Charge Solid Location and Transportation Problem (FCSLTP).

This problem seeks to integrate the facility location problem and the fixed charge solid transportation problem. These two problems are of different planning horizons ranging from long to short terms. Solving these problems independently might lead to suboptimal solutions. Therefore, an integrated solution will be necessary to prevent possible sub-optimality of the solutions. Furthermore, this problem is an extension of FCSTP, extended by the cost of facility location. Therefore, we implicitly classify this problem to fall in the NP-hard class of problems. We further propose a hybrid heuristic solution that uses the GA process to select a combination of feasible facility locations while allocation from the feasible locations is achieved using a constructive greedy heuristic. An improvement heuristic, which we have termed modified stepping stone algorithm, is used to further consolidate load distribution for cost reduction and improve the search for a better solution.

In order to test the effectiveness (objective function) and efficiency (solution time) of our hybrid GA method, we compare our results with the solutions provided by CPLEX, a commercial solver. Section 1 consist of significance, motivation and existing research on the variants of STP and solution methods. This section is concluded with a gap analysis as shown in Table 1 below. Section 2 consists of model formulation for the variants of FCSTP and FCSLTP. Section 3 provides a detailed explanation and illustration of the Hybrid heuristic considered. Preliminary experiments and computation study are presented in section4. The experimentation and results obtained between the hybrid heuristic and CPLEX are compared.

Section 6 summarizes the paper and provides future improvements to the hybrid heuristic.

Table 1: Problem Gap analysis

Selected Authors	Problem Characteristics					Solution Method
	Variable cost	Route fixed cost	Facility Location fixed cost	Conveyance constraint	Type	Class
Sani et al. (2017)	✓	✓	✗	✓	Lagrange Heuristic Vs CPLEX	Heuristics
Oyewole and Adetunji (2018)	✓	✓	✓	✓	Lagrange Heuristic Vs CPLEX	Heuristics
Das et al. (2019)	✓	✗	✓	✓	locate-allocate heuristic	Heuristics
This paper	✓	✓	✓	✓	Genetic Algorithm +greedy heuristic + Modified stepping stone Vs CPLEX	Hybrid of Meta-Heuristics and classical heuristics

2. Model Formulation

The FCSLTP is modelled as a mixed-integer linear programming problem consisting of m feasible sources or locations, n destinations or customers, and a conveyances or transport sources. Our FCSLTP basically differs from the FCSTP discussed in that location costs, location capacities, route costs and route capacities are simultaneously used in determining whether locations will be open or closed when servicing customers (Sanei et al., 2017). Figure 1 further describes the FCSLTP. Moreover, our model formulation and assumptions considered are similar to those presented and rehashed in this paper (Oyewole and Adetunji, 2018). They introduced the FCSLTP and attempted to solve the model using CPLEX and Lagrange relaxation heuristic. The CPLEX solution outperformed Lagrange relaxation heuristic. However, this paper considers another heuristics which is a hybrid of metaheuristic and classical heuristics and compares the solution obtained with that of CPLEX. The FCSLTP seeks to minimize total transportation and location costs by determining the optimal allocations from selected open locations through open routes via a set of conveyances. In order to ensure comprehension of our FCSLTP formulation, we present the formulation of the FCSTP as described (Sanei et al., 2017). Subsequently, we present the formulation of our FCSLTP.

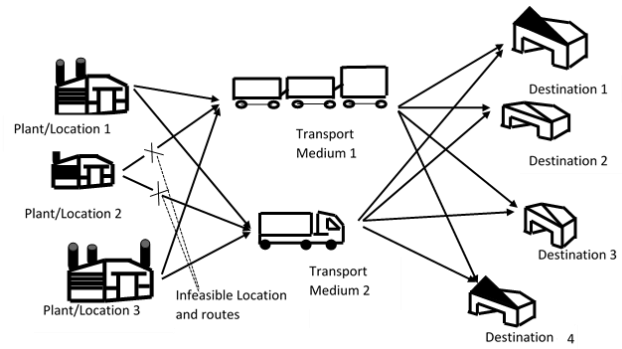

Figure 1: Illustration of FCSLTP

2.1 Model parameters

i: Index for sources or facility locations (warehouses, depots etc.)
j: Index for destinations (customers, other warehouses etc.)
r: Index for conveyances (or Transportation mediums)
m: Number of sources
n: Number of destinations
a: Number of conveyances
c_{irj} : Variable cost of shipment from source i through conveyance r to destination j.
S_i : Capacity at source i.
D_j : Demand at Destination j.
T_r : Capacity of conveyance r.
F_i : fixed charge for keeping a facility location open.
H_{irj} : Fixed cost (fixed charge) incurred for shipping from source i through conveyance r to destination j.

2.2 Decision Variables

x_{irj}: Quantity of products transported from source i through conveyance r to destination j.
y_i : Variable indicating which facility location is opened.
z_{irj} : Variable indicating which conveyance means is utilized en route (i, j).

2.3 Objective function for the FCSTP

$$\min \quad \sum_{i=1}^{m}\sum_{r=1}^{a}\sum_{j=1}^{n} c_{irj}x_{irj} \quad + \sum_{i=1}^{m}\sum_{r=1}^{a}\sum_{j=1}^{n} H_{irj}z_{irj} \tag{1}$$

Subject to

$$\sum_{r=1}^{a}\sum_{j=1}^{n} x_{irj} \leq S_i \qquad \forall\ i = 1 \dots m \tag{2}$$

$$\sum_{i=1}^{m}\sum_{r=1}^{a} x_{irj} = D_j \qquad \forall\ j = 1 \dots n \tag{3}$$

$$\sum_{i=1}^{m}\sum_{j=1}^{n} x_{irj} \leq T_r \qquad \forall\ r = 1 \dots a \tag{4}$$

$$x_{irj} \geq 0 \quad \forall\ i = 1 \dots m, r = 1 \dots a, \forall\ j = 1 \dots n, \forall \tag{5a}$$

$$z_{irj} = \begin{cases} 1 & x_{irj} > 0 \\ 0 & Otherwise \end{cases} \quad \forall\ i = 1 \dots m, \forall\ r = 1 \dots a, \forall\ j = 1 \dots n, \tag{5b}$$

Expression (1) is the objective function. The first term is the route variable cost per conveyance type and the second term is the route fixed-charge cost per conveyance type. constraint (2) is the supply capacity constraint ensuring no supply preference for selected locations. constraint (3) is the demand constraint to be met at each destination. Constraint (4) is the conveyance capacity. Constraint (5a) refers to the non-negativity constraint for the continuous variables and constraint (5b) refers to the binary constraints for the route fixed charge requirement.

2.4 Objective Function (minimum cost function) for FCSLTP:

Minimize (Z)=

$$\sum_{i=1}^{m} F_i\, y_i + \sum_{i=1}^{m}\sum_{r=1}^{a}\sum_{j=1}^{n} c_{irj}x_{irj} + \sum_{i=1}^{m}\sum_{r=1}^{a}\sum_{j=1}^{n} H_{irj}z_{irj} \quad (6)$$

Subject to

$$\sum_{r=1}^{a}\sum_{j=1}^{n} x_{irj} \leq S_i y_i \qquad \forall\ i = 1 \dots m \quad (7)$$

$$\sum_{i=1}^{m}\sum_{r=1}^{a} x_{irj} = D_j \qquad \forall\ j = 1 \dots n \quad (8)$$

$$\sum_{i=1}^{m}\sum_{j=1}^{n} x_{irj} \leq T_r \qquad \forall\ r = 1 \dots a \quad (9)$$

$$x_{irj} \geq 0 \qquad \forall\ i = 1 \dots m\ , \forall\ r = 1 \dots a\ , \forall\ j = 1 \dots n \quad (10a)$$

$$z_{irj} = \begin{cases} 1 & x_{irj} > 0 \\ 0 & Otherwise \end{cases} \quad \forall\ i = 1 \dots m\ , \forall\ ,, j = 1 \dots n\ , \forall\ r = 1 \dots a \quad (10b)$$

$$y_i = 0\ or\ 1 \qquad \forall\ i = 1 \dots m \quad (10c)$$

Expression (6) is the objective or the cost function, which we seek to minimize. The first term computes the total facility location cost, the second term computes the total route variable cost and the third term computes the route total fixed charge. Constraint (7) is the supply capacity constraint of each facility location or sources. It also ensures that capacities of closed facilities are not utilized. Constraint (8) is the demand constraint indicating the destination demands should be met. Constraint (9) is the conveyance capacity constraint. It ensures that capacities of selected conveyances are not exceeded. Constraint (10a) refers to the non-negativity constraint for the continuous variables. Constraint (10b) is binary indicating whether or not there is shipment using a conveyance along the particular route. Constraint (10c) is binary indicating whether a facility is opened or not.

3. Genetic Algorithm

The Genetic Algorithm (GA) is a multi-dimensional search strategy defined as a framework that imitates the evolutionary principle of nature to provide solutions to NP-hard combinatorial problems (Fernandes et al., 2014). The GA has also been viewed as a probabilistic or stochastic search technique due to the probability rates normally associated with the genetic operations involved in producing solutions during the search process. As noted by the successful implementation of the GA depends on the user-defined solution representations, initialization, genetic operations and terminating conditions (Jawahar et al., 2012; Perez-Salazar et al., 2015). The solution representation basically is concerned with how to encode and decode the feasible solution of the combinatorial problem taking part in the genetic operations. These feasible solutions are usually referred to as chromosomes. In addition, the representation of each individual variable type making up the chromosomes (i.e. genes) has to be properly captured. This is because optimization variables could either be continuous, binary or integer. The representation types used have been noted by several authors to determine how sensitive the GA will be in converging to the solution desired. The Genetic operations consist of the chromosome selection method, crossover operation and mutation operation used to ensure necessary diversity in the search process. The stages of the GA implementation include initialization, crossover, mutation and termination.

3.1 Initialization

The initialization conditions include the determination of the desired fitness function (objective function) for the GA procedure, chromosome representation, initial population size and the terminating condition of the GA, including the number of generations.

3.2 Crossover operation

The aim of the crossover is to generate and promote the replication of good solutions (chromosomes) while rejecting the bad ones. Before crossover is performed, chromosomes are selected using some selection probabilities. The roulette wheel technique is a popular selection technique used in literature to achieve the selection (Jawahar and Balaji, 2009; Ojha et al., 2010; Pérez-Salazar et al., 2015). The crossover operation ensures the reproduction of new offspring or children solution from parent solutions. Different cross over operation types have been discussed in the literature as noted (Jawahar and Balaji, 2009). These are either based on a single point or two-point crossover such as the partially mapped crossover and the ordinal mapped crossover.

3.3 Mutation operation

The mutation operation involves perturbation of some of the genes (variables) of a chromosome-based on some assigned probabilities known as the mutation rate. Genes are also randomly selected using a user-defined mutation rate. The mutation operation or gene replacement essentially gives the GA its power of arriving at other new solutions not possible with the crossover and have the potential of being better than existing solutions.

3.4 Termination

Terminating conditions usually involve the stopping criteria normally employed in optimization problems such as the number of desired iterations and optimization time desired. For the GA the number of generations employed can also be utilized as a stopping criterion.

3.5 Solution Representation

Choosing a suitable representation for the candidate solutions of the original problem has been considered by several authors to be based on the optimization problem structure and the ease of performing the genetic operations of the GA. The matrix and vector (binary) representation were discussed (Vignaux and Michalewicz, 1991). Priority based encoding was proposed by (Gen et al., 2006). This was to prevent likely infeasibility during genetic operations observed with the prüfer number technique of representing chromosomes discussed by (Gottlieb et al., 2001). Antony et al. [26], while discussing solutions to a m-number of sources and n-number of destinations FCTP, underscored the differences between the matrix, permutation, prüfer number and direct representation (Antony et al., 2011). The differences were based on the number of genes involved in the chromosomes. They showed the matrix representation as possessing the highest number of genes representing the transportation problem which is $m \times n$, while the prüfer number had the least i.e. $m + n - 2$. A hybrid chromosome representation that presents both the continuous and the binary variables of the original mixed-integer problems as an array was discussed (Perez-Salazar et al., 2015; Hiassat et al., 2017).

In this paper, a vector of binary numbers is used to represent the facility locations while a matrix of continuous numbers is used to represent the candidate solution, which is essentially the allocated quantity from the facility locations (sources) and to the points of demand (destinations). The facility location vectors are encoded as the GA chromosomes and manipulated through the various GA operations while the constructive greedy heuristics and improvement modified stepping stone heuristics work on the allocation matrix. Based on the result from the GA operations on the facility location vector (as shown in Figure 2) the allocations (shown in Figure 3) are made. A typical matrix representation used for a sample feasible solution to an original problem with 3-candidate facility locations, 4-demand destinations and two conveyances is shown in Figure 3. The facility location and route fixed charges are incurred when the continuous variable part of equation (1) are non-zero.

$i=1$	$i=2$	$i=3=m$
1	0	1

Figure 2: Sample chromosome representation for an FCSLTP with 3 sources

y_1 =1 S_1	x_{111} ,z_{111} =1	0	x_{113} ,z_{113} =1	0	T_1
	0	0	0	x_{124} ,z_{124} =1	T_2
y_2 = 0 S_2	0	0	0	0	T_1
	0	0	0	0	T_2
y_3 =1 S_3	x_{311} ,z_{311} =1	0	0	x_{314} ,z_{314} =1	T_1
		x_{322} ,z_{322} =1	0	0	T_2
	D_1	D_2	D_3	D_4	

Figure 3: Typical candidate solution representation for a 3-source, 4 – destination and 2- conveyance problem

3.6 Initialization

3.6.1 Fitness function

The fitness function to be used in the GA is the objective function of the original problem. This is same as equation (1) above.

3.6.2 Initial population and candidate feasible solution generation

Given the location fixed cost F_i of dimension (m), route fixed cost H_{irj} of dimension $(m \times a \times n)$, variable cost c_{irj} of dimension $(m \times a \times n)$, population size (p) and number of generation (g), the generation of the candidate feasible solutions to the original problem ($c_1 \dots c_p$) and the initial population of chromosomes are described below and illustrated in Figures 4 and 5 below respectively.

1. Random generation of the combination of facilities or locations that possess sufficient capacity to meet demand. We select y_i ($i = 1 \dots m$) such that the feasibility $\sum_{i=1}^{m} S_i\, y_i \geq \sum_{j=1}^{n} D_j$ is checked and uniqueness of each combination of facilities is ensured. A matrix $(m \times p)$ termed Pop_{Chrom} is created to store each feasible combination of the facilities. The matrix Pop_{Chrom} is also referred to as the population of chromosomes in this paper and represented in Figure 4 below.

2. Creation of a Relaxed Average Variable Cost (RAVC) matrix of dimension $(m \times a \times n)$. This is based on the integration of the route fixed cost, the variable cost of the problem and minimum of all capacities and it is used to allocate capacities. This is similar to the least equivalence variable cost discussed by Jawahar, Balaji [22]. The $RAVC$ is stated as:

$$RAVC\ (irj) = \frac{Route\ fixed\ cost\ (H_{irj})}{\min(S_i, D_j, T_r)} + Variable\ cost\ (c_{irj}) \qquad (11)$$

3. Creation of the matrix of candidate feasible allocation (Illustrated with Figure 5). We created a three-dimension matrix of dimension $(ma \times n \times p)$. This is called the candidate feasible solution allocation matrix and termed CFS_p. The procedure for creating CFS_p is stated below.

   (a) From the earlier Pop_{Chrom} of feasible combinations of facilities (chromosomes), select each feasible chromosome (m) from the (p) rows of population.

   (b) Compute the RAVC as stated above to obtain the matrix$(m \times a \times n)$.

   (c) Apply the constructive greedy heuristic (illustrated with Figure 6) to make the initial allocation. The greedy heuristic utilizes the m rows of the matrix obtained in step (3a) and the RAVC computed in step (3b) to allocate into the first layer of CFS_p with dimension $(ma \times n \times 1)$. This is termed CFS_1

   (d) Use the improvement heuristic (the modified stepping stone algorithm) (illustrated in Figure 7). This is based on the actual route fixed cost and variable cost matrix, to improve allocations obtained in step(3c) above. This gives the final allocation for the initial candidate feasible CFS_1.

   (e) Repeat step (3a) to (3d) for all the candidate feasible solution ($c_1 \dots c_p$) to obtain CFS_p.

4. Computation of the candidate feasible solution fitness function using the actual cost parameters and the allocation of CFS_p obtained in step (3e) above.

$i=1$	$i=2$	...	$i=m$	$m \times p$
1	0	...	1	$pop = 1$
1	0	...	0	$pop = 2$
		⋮		⋮
0	1	...	1	$pop = p$

Figure 4: Sample populations of Chromosomes $(m \times p)$

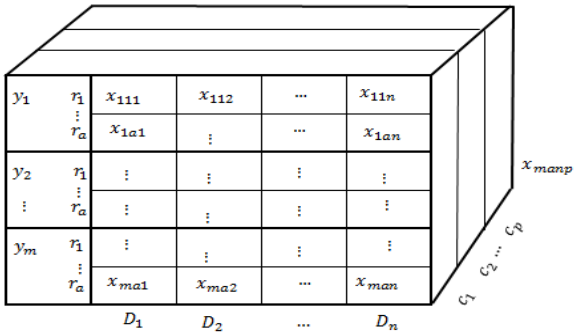

Figure 5: Candidate feasible solution allocation matrix($\mathbf{CFS_p}$)

3.6.3 Greedy heuristic

This utilizes the RAVC to allocate demand such that the source, transport and demand capacities are not exceeded. This is further illustrated in Figure 6.

Greedy allocation heuristic

For every entry of demand, $j = 1{:}n$,
find the minimum RAVC from $(k = 1 : ma)$ as (k, j)
locate corresponding source $i = 1{:}m$ and $r = 1{:}a$ to selected (k, j)
While demand $(j) > 0$
If $T_r > 0$ and $S_i > 0$
x_{irj} = allocation = min (S_i, D_j, T_r)
Subtract x_{irj} from D_j
Subtract x_{irj} from S_i
Subtract x_{irj} from T_r
Else
Move to the next minimum RAVC (k, j)
End if
Update j: $j = j + 1$
End for

Figure 6: Greedy heuristic to populate Initial solution

3.6.4 Improvement Heuristic (Modified stepping stone method)

The modified stepping stone method is done in order to check for possible cost savings through route fixed cost and variable cost trade-off by either eliminating route fixed costs and/or possible reduction in variable costs subject to capacity allocation. A set of acronyms are defined below in the modified stepping stone method for comprehension purposes and presented below.

i, r *and* j are already defined under section 2.1
(m, n, a) is as stated in the original problem in section 2.1
Define source indices $(i \text{ and } u)$: $i < u \leq m$
Define destination indices $(j \text{ and } p)$: $j < p < n$
Define conveyance indices $(r \text{ and } v)$: $r < v < a$
x_{irj} => variable allocation at positon (irj). (Similarly for $x_{irp}, x_{uvp}, x_{uvj}$)
H_{ijr} => Route fixed cost at position (irj).(Similarly for $H_{irp}, H_{uvp}, H_{uvj}$)
min_alloc => Minimum of allocation.
variable_cost change at position (irj)= $(c_{irj} + c_{uvp}) - (c_{irp} + c_{uvj})$.
(based on $x_{irj}, x_{irp}, x_{uvp}, x_{uvj}$ and Illustrated in Figure 7)

The improvement heuristic is illustrated in Figure 7, while an Illustration of the selection of variables for the improvement heuristic(modified stepping stone consolidation) is presented in Figure 8 .

Improvement Heuristic (Modified stepping stone method)

For every source, transport means $(i = 1, r = 1)$ *to* $(i = m, r = a)$
Source = source 1
For every destination $j = 1{:}n-1$, if $x_{irj} > 0$
Source, Destination = source1, dest 1
(# Source, Destination combination is explained in Figure 7 below #)
If for any destination $p > j$, $x_{irp} > 0$
Source, Destination = source 1, dest 2
If for any source $(u > i \text{ and } v >= r)$ OR source $(u >= i \text{ and } v > r)$, $x_{uvp} > 0$
Source, Destination = source 2, dest 2
If $x_{uvj} > 0$
Source, Destination = source 2, dest 1
Find $min\_alloc = min(x_{irj}, x_{irp}, x_{uvp}, x_{uvj})$
If H_{irj} < variable_cost change (check 1) THEN
(consolidation step1)
$x_{irj} = x_{irj} - min\_alloc$
$x_{uvp} = x_{uvp} - min\_alloc$
$x_{irp} = x_{irp} + min\_alloc$
$x_{uvj} = x_{uvj} + min\_alloc$
(Repeat check1 for fixed costs positions H_{irp}, H_{uvp} and H_{uvj} and apply pattern of consolidation step1 if true)

ELSE If H_{irj} > variable_cost change (check 2)THEN
(consolidation step2)
$x_{irp} = x_{irp} - min\_alloc$
$x_{uvj} = x_{uvj} - min\_alloc$
$x_{irj} = x_{irj} + min\_alloc$
$x_{u\,vp} = x_{u\,vp} + min\_alloc$

(Repeat check2 for fixed cost positions H_{irp}, H_{uvp} and H_{uvj} And apply pattern of consolidation step2 if true)

Else (no improvement for cost position irj)

Figure 7: Improvement heuristic **(**modified stepping stone procedure).

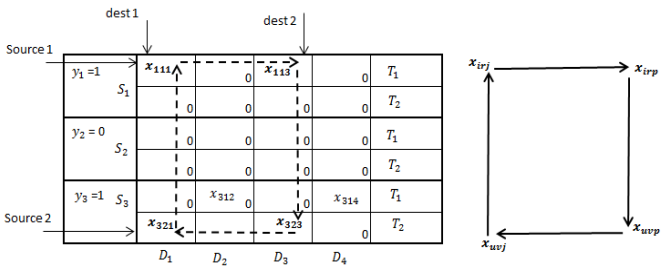

Figure 8: Selection of variables for load consolidation

3.7 Generation of New Population

The generation of new chromosomes is discussed in this section. We put an emphasis on the best fit solution over the weak solutions during the crossover and mutation operations as also indicated by in their solution (Jawahar and Balaji, 2009). As described in section 3.2 the generation of the initial population is done by generating a combination of facilities such that $\sum_{i=1}^{m} S_i\, y_i \geq \sum_{j=1}^{n} D_j$ The random search is implemented for the binary facility location term in this paper and stored in a matrix of dimension $(m \times p)$ as shown in Figure 4 above.

3.7.1 Inputs for new population

The new population generation function takes as input the following:

a) The parent matrix $(m \times p)$ generated in section 3.2
b) A vector of sort index of chromosomes $(m \times p)$ in increasing order of cost for each chromosome (dimension$p$).
c) The crossover rate $(cross\ rate)$
d) The mutation rate $(mut\ rate)$
e) The source and demand capacities corresponding to the matrix $(m \times p)$.

3.7.2 Genetic operations procedure

The matrix of the old facilities opened (parent) contains p chromosomes, each chromosome being a set of binary values indicating which supply points were opened or closed. This matrix was crossed over and mutated to create a new population on which the allocation and improvement heuristics were applied. This cycle was repeated until the numbers of generations (g) were completed.

This procedure is described below:

1. Determine the number of chromosome of the old population to keep from the crossover rate $(cross\ rate)$.
2. Populate the discarded chromosomes to build up a matrix of a new population of size $(m \times p)$ using the procedure below.
   a. Copy the retained chromosomes into the relevant positions in the new population matrix, keeping the least cost chromosome in position 1.
   b. Use rank based roulette wheel selection (as shown beloow in Figure 9) to select the two mating chromosomes among the retained chromosomes. The rank-based roulette wheel selects two chromosomes to be used for crossover from the chromosomes retained from the population. It receives as input a population of chromosomes ranked based on fitness function from the best ranked in the first position to the worst ranked in the last position.
   c. Randomly generate the crossover point for the mating chromosomes as described below.
   d. Perform crossover (described blelow in Figure 10) and store the two new offsprings in the next two positions in the new population matrix.
   e. Repeat step (d) until the new population matrix is fully constructed
3. Use the mutation rate to determine the number of genes to mutate by flipping the binary value (0 *to* 1 *or* 1 *to* 0). This is described in section 3.3.4 below.
4. Randomly generate the two index positions to mutate in the new matrix and flip the gene in the location while preserving the least cost gene in position 1 unchanged
5. For every chromosome in position 2 till the last, check for feasibility (section 3.3.5)
   a. If the chromosome is not feasible, randomly locate a position that is closed and open until the chromosome has a number of opened sites that is feasible for demand allocation
6. Once new allocation matrix is complete, pass matrix to the greedy algorithm to allocate demand and consolidate the allocation using the modified stepping stone algorithm
7. Repeat all steps until the number of generation is complete.

Rank based roulette selection

rank = ranking of chromosome in the population
p = population size
cumProb = sum of probability up until the current member of the population, initialised to zero
sumRank = sum of the rank of all members initialised to zero
chrom 1 = First chromosome selected for crossover
chrome 2 = Second chromosome selected for crossover

for all members of population,
  sumRank = sumRank + rank of chromosome
end for

for all members of population,
  cumProb = cumProb + ((p – rank + 1) / sumRank)
end for

Generate the mating chromosomes
  number = random between 0 and 1
  start from first member of population
    while number =< cumProb
      then chrom 1 = current chromosome
      go to next member
    end while
  Repeat for chrom 2 what was done for chrom 1

Return chrom 1 and chrom 2

Figure 9: Rank based roulette selection

3.7.3 Crossover operation

For the crossover operation, two chromosomes are selected as a pair and a crosspoint is randomly generated for the pairwise interchange. A description of the algorithm for this is presented in the Figure 10.

Given a pair of chromosomes to undergo crossover operation
first chromosome = chrom 1
second chromosome = chrom 2
length of chromosome = chromLength ,
crossover point = crossPoint = random integer between 1 and chromLength

For chrom 1 in the pair,
  Copy gene from chrom 2 from crossPoint to the end.
  Put into the same position in chrom 1.
  Assign to offspring 1.

For chrom 2 in the pair,
  Copy gene from chrom 1 starting from position 1 to position crossPoint
  Put into the same position in the chrom 2
  Assign to offspring 2.

Return offsprings 1 and offspring 2

Figure 10: Description for Crossover operation

3.7.4 Mutation operation

In this we discuss the mutation operation performed which is similar to the description presented in section 3. The number of genes to mutate in the population is determined based on a mutation rate. Following this, selected genes are interchanged with other selected genes in the population. This is further described in Figure 11.

p = number of row of matrix, equal to the population size.
numSource = number of genes in a chromosome, total equal to the number of sources (m).
mutPercent = percentage of matrix genes to mutate.
mutNum = numbers of genes to mutate = (p) × (numSource) × mutPercent

number 1 = Generate a random number between 1 and numSource.
number 2 = Generate a random number between 2 and p.

for 1 till mutNum
  flip the gene in position (number 1, number 2) of the matrix
endfor

Figure 11: Description of mutation operation

3.7.5 Check for feasibility

This section checks every chromosome representing a combination of opened sites out of all possible sources to ensure feasibility. It accepts as inputs the vector of demand at each destination, vector of supply capacity at each source and the matrix of opened sites. This is presented in Figure 12.

sumDem = sum of all destination demands (from demand vector).
sumOpenedCap = sum of capacities of all opened sites, based on a chromosome
numSource = number of source sites available, equal to the number of columns of the matrix and also equal to the length of a chromosome(m).
population size (p) = equal to the number of rows of matrix.

for all chromosomes (rows)
  while sumproduct(current chromosome vector and source capacity vector) < sumDem
    randomly flip one of the zeros to 1
  end while
end for

Figure 12: Description of chromosome feasibility

Figure 13 below shows a summary of the working procedure of the HA

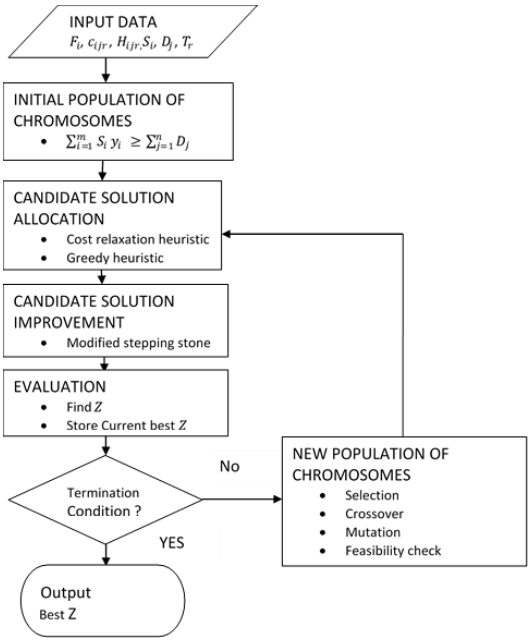

Figure 13: Hybrid Algorithm flow chat

4. COMPUTATION STUDY

We did the computational study in two stages. The first stage is preliminary experimentation while the second is the main experimentation. The preliminary experimentation was performed to obtain necessary parameters to effectively implement the HA. In addition, it was done to identify the most influential parameters of the HA by observing the relative effectiveness. The parameters such as the population size (***p***), number of generations (***g***), crossover rate (***cross rate***) and mutation rate (***mut rate***) implemented, have been shown in the literature to affect the convergence of a GA solution. Studies in literature have shown that important parameter settings for a GA to be based on tuning the population size and number of generations (Ho and Ji, 2005; Fernandez wt al., 2014; Guo et al., 2017). In the main experimentation, we assess the quality of solutions of the HA and CPLEX. This is based on measures of performance described in section 5.0.

4.1 Preliminary experimentation

In this section the population size, number of generation, crossover rate and mutation rate are varied. The problem sizes have been stated in the order of ($m \times n \times a$). Where m = number of sources (or locations), n = number of destinations or demand points and a = number of transport sources or conveyances. In conducting parameter tunings for our HA,

problem sizes (5×8×2, 7×10×2, 9×12×2 and 15×30×2) which represent a sample of small to medium-sized problems comprising our test data were selected. Population sizes considered for the smaller problem were smaller than the medium sized problem to account for the uniqueness of solution required to populate each population size. In both instances initial crossover and mutation rates were randomly selected and kept constant while the population size and number of generations were varied in an increasing order. The population size and number of generations that showed a quick convergence were retained. Similarly, the best performing population size and number of generations were kept constant while the crossover rate and mutation rate were progressively varied. Tables 2 shows the parameter variations and convergence of the test problem sizes utilized. The values of the minimum cost obtained for the first and Last iterations are recorded as MinCost(first) and MinCost(last) respectively.

Table 2: Paramter tuining results.

Problem Size	Problem Characteristics1	Problem Characteristics2	MinCost first	MinCost Last
5X5X2	p	g		
	10	8	27689	27058
	20	20	27024	27006
	30	50	27006	27006
	cross rate	*mut rate*	$p = 30$	$g = 50$
	0.7	0.3	28059	27006
	0.3	0.3	27342	27006
	0.3	0.1	27024	27006
8X8X2	p	g		
	10	8	27696	26782
	20	20	27266	26782
	30	50	26810	26782
	cross rate	*mut rate*	$p = 30$	$g = 50$
	0.7	0.3	26810	26782
	0.3	0.3	26782	26782
	0.3	0.1	26782	26782
9X12X2	p	g		
	20	8	37694	36478
	50	50	37414	35826
	50	100	36611	35826
	cross rate	*mut rate*	$p = 30$	$g = 50$
	0.7	0.3	37004	35826
	0.3	0.3	37401	35826
	0.3	0.1	36608	35826
15X30X2	p	g		
	20	8	77240	71408
	50	50	75100	70927
	50	100	73084	70927
	cross rate	*mut rate*		
	0.7	0.3	73915	70927
	0.3	0.3	75150	70927
	0.3	0.1	75085	70927

Results obtained indicate that the parameter combinations all converged to the same minimum cost except in the cases of (=10 and g =8), (p =20 and g =8) (p =10 and g =8). Some parameter combinations obtained lower minimum cost from the initial generation (iteration). This possibly could indicate a quick convergence when using such parameter combinations for the HA. For the test problem sizes (5×8×2) and (7×10×2), the results showed that the population size (30), number of generations (50), crossover rate (0.3) and mutation rate (0.1) converged rather quickly for the minimum cost value compared to other parameters used. Results of the problem sizes (9×12×2) and (15×30×2) showed that population size (50), number of generations (100), crossover rate (0.5) and mutation rate (0.1) converged more quickly than with other parameters. In summary, the population size and number of generations seemed to be very effective in determine the final minimum cost value obtained.

4.2 Data Generation for experimentation

A modification to experimental data was used to test the different solution methods (Sanei et al., 2017). We extended their benchmark data to capture the cost of facility location which was not considered in their model. For the facility location cost, we have used the method of generating facility location cost instances from the supply capacities considered in facility location literature as used (Gadegaard et al., 2017; Fishetti et al., 2016; Guastaroba and Speranza, 2014). In this method, the facility location cost is calculated using $F_i = U(0,90) + \sqrt{S_i}\ \ U(100,110)$. Uniformly distributed data randomly generated as integers in a unit square coordinate U [a, b] were considered for the experiments. The letter "a" refers to the lower cost limit and "b" is termed the upper cost limit. A total of 45 problems instances across 9 different problem sizes were considered for the main experimentation. We have termed problem size number (1) to (4) and (5) to (9) as small and medium sized problems respectively. A summary of the Problem sizes considered and the parameters used for data generation are given in the Tables3 and 4 respectively.

Table 3: Parameter distribution used for computations

Problem Size No.	Problem Size $\boldsymbol{m \times n \times a}$	No of instances
1	5×5×2	5
2	5×8×2	5
3	7×10×2	5
4	8×8×2	5
5	9×12×2	5
6	10×10×2	5
7	10×20×2	5
8	15×30×2	5
9	30×30×2	5

Table 4: Parameter distribution used for experimentation

Parameter Distribution
$\boldsymbol{S_i}$ **U(200, 400)**
$\boldsymbol{D_j}$ **U(50, 100)**
$\boldsymbol{T_r}$ **U(800, 1800)**
$\boldsymbol{c_{ijr}}$ **U(20, 150)**
$\boldsymbol{H_{ijr}}$ **U(200, 600)**
$\boldsymbol{F_i = \mathrm{U}(0,90) + \sqrt{S_i}\, U(100,110)}$
$\boldsymbol{M_{ijr} = \min(S_i, D_j, T_r)}$

4.3 Solution methods

We have utilized the IBM CPLEX as a solution method in this paper. IBM CPLEX utilizes the conventional branch and cut algorithm and also implements a dynamic search algorithm. According to the IBM reference manual, the dynamic search algorithm basically uses the Branch and Cut algorithm with heuristics for quick termination of some nodes explored as the optimization technique (Studio, 2016). It is also indicates that the dynamic search algorithm consists of LP relaxation, branching, cuts and heuristics. At the default settings, CPLEX decides whether to provide solution using the conventional branch and cut or the dynamic search algorithm based on the model formulation (Studio, 2016). We have used the IBM CPLEX 12.8 dynamic search as a solution method to the original problem. This can imply a possible conventional Branch and Cut or dynamic search could be used by CPLEX to find a solution. Our HA was coded using Matlab 7.4.0. Based on the results from the preliminary experimentation, the HA was computed with population size (30), number of generations (50), crossover rate (0.3) and mutation rate (0.1) for the small problem sizes. For The medium problem sizes, the HA was computed with population size (50), number of generations (100), crossover rate (0.5) and mutation rate (0.1). The Solution methods were implemented on a Windows 8.1 Laptop with 6GB RAM and a processor speed of 2.5GHz

5. Experimentation and Discussion of Results

The performance of each solution method was determined under the following test categories.

a) A preliminary experimentation to determine the HA parameters for the main experimentation. This was computed in section 4.1 above.

b) Mean of each problem size. This was calculated based on effectiveness and efficiency of each solution method. Our mean value was expressed with the notations: $CPLEX_{mean}$ and HGA_{mean}.

$$CPLEX_{mean} = \frac{\sum_{i=1}^{5} CPLEX\ SM_i}{5} \quad or$$

$$HGA_{mean} = \frac{\sum_{i=1}^{5} HGA\ SM_i}{5}$$

$$i = index\ of instance\ number\ of\ each\ problem\ size$$

$$CPLEX\ SM \Rightarrow CPLEX\ minCost\ or\ CPLEX\ time$$
$$HGA\ SM \Rightarrow HGA\ minCost\ or\ HGA\ time$$

The mean values give an indication of the problem size effectiveness or efficiency

c) Instance and mean gap computation. This was also calculated based on the effectiveness and efficiency of each solution method. The Instance and mean gap computation were computed as percentages respectively. These were expressed with the notations % gap_i and % gap_{mean} respectively.

$$\%\ gap_i = \left(\frac{HGA\ SM_i - CPLEX\ SM_i}{CPLEX\ SM_i}\right) \times 100$$

$$\%\ gap_{mean} = \left(\frac{HGA\ SM_{mean} - CPLEX\ SM_{mean}}{CPLEX\ SM_{mean}}\right) \times 100$$

$$i = index\ of instance\ number\ of\ each\ problem\ size$$

Other notation are as previously indicated in (b) above.

We state that % gap_i or % gap_{mean} values obtained can either be zero, positive number or negative number. A zero value indicates equivalent performance from both methods. A positive value indicates that CPLEX obtained better results. A negative value indicates that the HA obtained better results. The results obtained for each of the problem instances based on our defined measures of effectiveness, and efficiency are presented in Table 5 below. The values obtained for the individual cases and the categorical averages are presented next. We start with the instance observations in Table 5, followed by the problem size averages in Table 9.

Table 5: Problem instance effectiveness, efficiency and % gap_i

Problem Size	Instance no	***CPLEX minCost***	***HGA minCost***	***CPLEX time***	***HGA time***	% **gap_i** (Min Cost)	% **gap_i** (Time)
5X5X2	1	18417.92	18417.92	1.14	0.13	0.0%	-88%
	2	18801.22	18534.22	1.12	0.13	-1.4%	-88%
	3	15369.09	21830.55	1.12	0.13	42.0%	-89%
	4	18828.75	23005.75	1.13	0.13	22.2%	-89%
	5	17090.47	16662.33	1.13	0.12	-2.5%	-89%
5X8X2	1	31320.91	31320.91	3.06	0.17	0.0%	-94%
	2	26301.38	30143.11	3.00	0.17	14.6%	-94%
	3	24209.66	31825.38	3.02	0.16	31.5%	-95%
	4	25539.24	25539.24	3.05	0.17	0.0%	-94%
	5	25864.92	25798.92	3.01	0.17	-0.3%	-94%
7X10X2	1	34676.97	34806.92	4.71	0.20	0.4%	-96%
	2	34760.30	33612.17	4.57	0.20	-3.3%	-96%
	3	32667.86	34587.04	4.53	0.23	5.9%	-95%
	4	36182.85	35841.88	4.72	0.20	-0.9%	-96%
	5	31334.72	31214.63	4.54	0.20	-0.4%	-96%
8X8X2	1	26952.66	26902.27	2.96	0.16	-0.2%	-95%
	2	26434.64	25816.64	2.97	0.18	-2.3%	-94%
	3	23830.67	23830.67	2.94	0.18	0.0%	-94%
	4	27032.24	27225.24	2.93	0.16	0.7%	-95%
	5	25864.92	25316.20	2.96	0.17	-2.1%	-94%
9X12X2	1	39844.21	37944.21	6.52	1.09	-4.8%	-83%
	2	35472.49	36024.36	6.61	1.06	1.6%	-84%
	3	39697.67	43278.10	6.88	1.11	9.0%	-84%
	4	41005.10	42701.06	6.70	1.04	4.1%	-84%
	5	40125.84	40749.84	6.53	1.03	1.6%	-84%
10X10X2	1	30471.92	30411.76	4.67	0.89	-0.2%	-81%
	2	28252.39	28252.39	4.82	0.99	0.0%	-79%
	3	35462.65	34160.77	4.63	0.88	-3.7%	-81%
	4	27648.69	27648.69	4.67	0.90	0.0%	-81%
	5	29211.92	29496.92	4.72	0.95	1.0%	-80%
10X20x2	1	64045.09	65361.83	18.00	1.77	2.1%	-90%
	2	57654.61	58505.61	17.40	1.63	1.5%	-91%
	3	62724.33	66508.88	18.19	1.56	6.0%	-91%
	4	63389.91	67396.36	18.12	1.79	6.3%	-90%
	5	68527.51	74904.08	17.90	1.81	9.3%	-90%
15X30X2	1	82136.32	93036.10	54.26	2.62	13.3%	-95%
	2	85684.45	87040.73	56.17	2.56	1.6%	-95%
	3	85515.94	89097.15	55.61	2.59	4.2%	-95%
	4	81937.51	81588.46	55.20	2.71	-0.4%	-95%
	5	83554.61	89640.24	55.46	2.66	7.3%	-95%
30X30X2	1	76649.71	75140.45	59.25	2.83	-2.0%	-95%
	2	81897.82	82452.20	54.88	2.94	0.7%	-95%
	3	81475.84	81725.59	64.07	2.92	0.3%	-95%
	4	78941.52	79702.55	57.16	2.74	1.0%	-95%
	5	88615.06	89680.65	57.33	2.95	1.2%	-95%

5.1 Observations on CPLEX and HA minimum cost results

Based on our effectiveness measure of performance, CPLEX should expectedly obtain better results than the HA. This is because of the conventional Branch and Cut implemented by CPLEX. The Branch and Cut has been noted to be an exact algorithm that can generate optimal solutions like Branch and Bound (Wolsey et al., 1998). Since CPLEX uses the branch and cut algorithm, we can expect that the solution provided by CPLEX should be at least as good as that provided by our HA algorithm. However, the use of some other search heuristics to speed up the convergence of CPLEX in its dynamic search function as indicated can lead to some less superior results than the HA (Studio, 2016). This is possible because of some nodes that could have been fathomed away, especially in instances where there are many nodes at a given search level with close possible final solution in their exploration, whose exploration may be considered not to hold so much promise but can significantly increase the time of convergence of the solution. We further present an instance of problems solved (Table 6 below) and show the actual allocations of randomly selected instances (Tables 7 and 8) where the HA seemed to obtain marginally better results than the CPLEX. The Feasibility of both solutions method is also observed.

Table 6: A Sample 5X5X2 Problem

F_i	1963.966	1883.348	2040.107	1682.118	2103	
S_i	311	340	378	227	361	
D_j	89	57	90	71	56	
T_r	1736	1488				
	c_{i1j}					
i	$j=1$	$j=2$	$j=3$	$j=4$	$j=5$	
1	58	36	44	146	70	$r=1$
2	100	42	98	89	97	
3	83	70	118	50	114	
4	90	144	93	49	26	
5	142	45	133	98	24	
i	c_{i1j}					
1	101	25	96	84	94	$r=2$
2	45	33	74	97	104	
3	27	55	127	32	67	
4	51	114	44	45	22	
5	132	22	109	87	83	
i	H_{i1j}					
1	330	435	268	223	357	$r=1$
2	294	272	591	460	572	
3	441	254	508	290	214	
4	263	375	586	378	436	
5	559	559	231	564	295	
i	H_{i2j}					
1	562	300	227	294	460	$r=2$
2	578	515	546	354	253	
3	334	372	232	303	565	
4	456	425	408	258	393	
5	384	331	379	414	236	

Table 7: Allocations obtained by CPLEX

i						
1	0	0	0	0	0	
2	0	0	0	0	0	
3	0	0	0	0	0	$r=1$
4	0	0	0	0	56	
5	0	0	0	0	0	
1	0	0	0	0	0	
2	0	0	0	0	0	
3	89	57	0	71	0	$r=2$
4	0	0	90	0	0	
5	0	0	0	0	0	
j	1	2	3	4	5	

Table 8: Allocations obtained using HA

i						
1	0	0	0	0	0	
2	0	0	0	0	0	
3	0	0	0	0	0	$r=1$
4	0	0	0	0	0	
5	0	0	0	0	0	
1	0	0	0	0	0	
2	0	0	0	0	0	
3	89	57	0	71	0	$r=2$
4	0	0	90	0	56	
5	0	0	0	0	0	
j	1	2	3	4	5	

We define limits of significant difference in effectiveness performance as about 5% based on the popular alpha level applied on many empirical tests. This implies that any difference in values between the results posted by CPLEX and HA that is in this range is probably not significant enough. If we follow this approach it can be seen in Table 5 above that the instances tend to show less significant differences with increase in problem sizes, while more significant differences are observed with the smaller problem sizes. In addition, the percentage difference in solution time (% gap $_i$ (Time) across the problem instances indicates that the HA solution seems much faster than CPLEX.

Table 9: Mean effectiveness, mean efficiency and % gap $_{mean}$

Problem Size	$CPLEX_m$ (Min Cost)	HGA_m (Min Cost)	$CPLEX_m$ (Min Cost)	HGA_m (Time)	% gap $_m$ (Min Cost)	% gap $_m$ (Time)
5X5X2	17701.49	19690.16	1.13	0.13	11.2%	-89%
5X8X2	26647.22	28925.51	3.03	0.17	8.5%	-95%
7X10X2	33924.54	34012.53	4.61	0.21	0.3%	-96%
8X8X2	26023.03	25818.20	2.96	0.17	-0.8%	-94%
9X12X2	39229.06	40139.52	6.65	1.07	2.3%	-84%
10X10X2	30209.51	29994.11	4.70	0.92	-0.7%	-80%
10X20x2	63268.29	66535.35	17.92	1.71	5.2%	-90%
15X30X2	83765.77	88080.54	55.34	2.63	5.2%	-95%
30X30X2	81515.99	81740.29	58.54	2.88	0.3%	-95%

The group results as shown in Table 9 above also indicate that there seems not to be any significant difference in effectiveness as the problem size increases while efficiency seems to favour HA. Figure 14 below shows a plot of the solution time of the HA and CPLEX as the problem size increases. The increase in solution time trend indicates possibility of the HA obtaining solutions faster than CPLEX when interpreted as probably being equivalent as stated earlier.

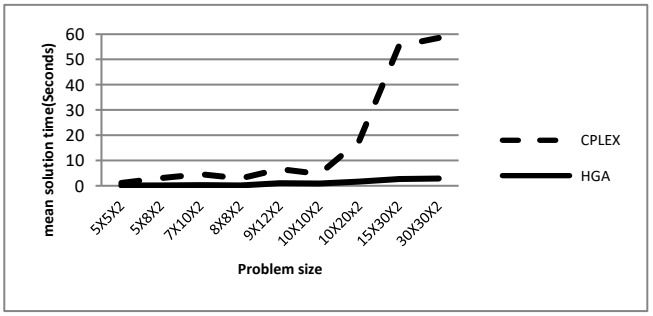

Figure 14: Computation time between CPLEX and HA per problem size

In Figure 15 we show a representation of the number of times each solution method obtained comparable results for each problem size. By comparable results we imply the results obtained by HA are within a neighborhood of between (0% and -5%) of CPLEX values. In Figure 15, we also show a representation of results of CPLEX that were significantly better than HA. By being significantly better we imply the computations % gap $_i$ (Min Cost) greater than 5%. These analyses were done for the minimum cost computation (effectiveness).

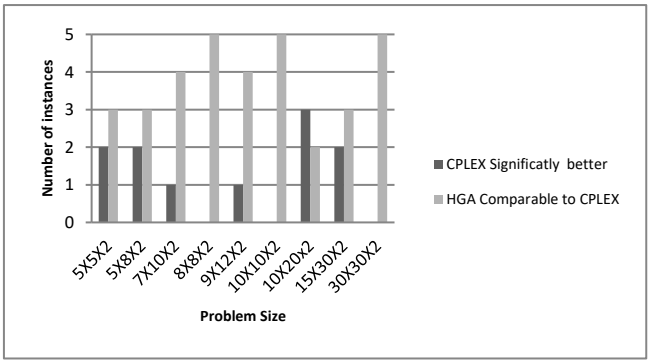

Figure 15: Instances CPLEX obtained comparable and significantly better results than HA

A summary of the results presented, showed that CPLEX seems able to obtain better results than the HA most of the times. Nonetheless, CPLEX could become computationally intensive as problem size increases and may return good values in an exponential time when larger sized problems are considered. Therefore, the HA which may not guarantee the best solutions most of the times but converges faster with good performances for many problem instances can be of significant value for solving the FCSLTP.

6. Conclusion and Future Direction

An optimization problem that integrates facility location and the fixed charge solid transportation problem was considered in this paper. This problem was termed Fixed Charge Solid Location and Transportation problem (FCSLTP). In order to solve this problem, solutions from CPLEX commercial solver and our Hybrid Algorithm (HA) were considered. Our HA utilizes the Genetic Algorithm framework to generate a population of feasible facility locations, while a greedy heuristic which uses cost relaxations implements the load allocation. After the allocations are done, an improvement heuristic is used to further search the solution space for better results. Some measures of performance such as mean, percentage instance and mean gap were used to assess the HA and the CPLEX solutions. The solution time and mean solution time were also used to assess both solution methods.

The HA demonstrated a competitive performance in obtaining solutions within the neighborhood of CPLEX values based on our effectiveness measure. In addition, the solution time of the HA seems much faster than CPLEX through all the problem sizes considered, and this even more so as the problem size increases. However, overall effectiveness results indicate that CPLEX can obtain results comparable to and sometimes much significantly better than HA. This however could be computationally intensive for CPLEX as problem size increases. Possible extensions to the HA include using a modified stepping stone that can search the non-basic positions of the load allocations. Furthermore, the GA or other metaheuristics could also be used to perturb the load allocations in search of better results. The reduced computation time of the HA makes it suitable as a hybrid heuristic to other solution methods to obtain better results.

References

Antony, K., Raj, D., Rajendran, C., 2011. A Hybrid Genetic Algorithm for Solving Single-Stage Fixed-Charge Transportation Problems. Technology Operation Management, 2 (1), Pp. 1.

Balaji, A., Nilakantan, J.M., Nielsen, I., Jawahar, N., Ponnambalam, S., 2019. Solving fixed charge transportation problem with truck load constraint using metaheuristics. Annals of Operations Research, 273 (1-2), Pp. 207-236.

Calvete, H.I., Galé, C., Iranzo, J.A., 2016. An improved evolutionary algorithm for the two-stage transportation problem with fixed charge at depots. OR spectrum, 38 (1), Pp. 189-206.

Chen, L., Peng, J., Zhang, B., 2017. Uncertain goal programming models for bicriteria solid transportation problem. Applied Soft Computing, 51, Pp. 49-59.

Das, S.K., Roy, S.K., Weber, G.W., 2019. Heuristic approaches for solid transportation-p-facility location problem. Central European Journal of Operations Research, Pp. 1-23.

Fernandes, D.R., Rocha, C., Aloise, D., Ribeiro, G.M., Santos, E.M., Silva, A., 2014. A simple and effective genetic algorithm for the two-stage capacitated facility location problem. Computers & Industrial Engineering, 75, Pp. 200-208.

Fischetti, M., Ljubić, I., Sinnl, M., 2016. Benders decomposition without separability: A computational study for capacitated facility location problems. European Journal of Operational Research, 253 (3), Pp. 557-569.

Gadegaard, S.L., Klose, A., Nielsen, L.R., 2017. An improved cut-and-solve algorithm for the single-source capacitated facility location problem. EURO Journal on Computational Optimization, Pp. 1-27.

Gen, M., Altiparmak, F., Lin, L., 2006. A genetic algorithm for two-stage transportation problem using priority-based encoding. OR spectrum, 28 (3), Pp. 337-354.

Genova, K., Guliashki, V., 2011. Linear integer programming methods and approaches–a survey. Journal of Cybernetics and Information Technologies, 11 (1).

Ghassemi, T.F., Hashemi, Z., 2018. Three hybrid GAs for discounted fixed charge transportation problems. Cogent Engineering, 5 (1), Pp. 1463833.

Gottlieb, J., Julstrom, B.A., Raidl, G.R., Rothlauf, F., 2001. Prüfer numbers: A poor representation of spanning trees for evolutionary search. In: Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation, pp. 343-350. Morgan Kaufmann Publishers Inc.

Guastaroba, G., Speranza, M.G., 2014. A heuristic for BILP problems: The Single Source Capacitated Facility Location Problem. European Journal of Operational Research, 238 (2), Pp. 438-450. doi:10.1016/j.ejor.2014.04.007

Guo, P., Cheng, W., Wang, Y., 2017. Hybrid evolutionary algorithm with extreme machine learning fitness function evaluation for two-stage capacitated facility location problems. Expert Systems with Applications, 71, Pp. 57-68.

Halder, S., Das, B., Panigrahi, G., Maiti, M., 2017. Some special fixed charge solid transportation problems of substitute and breakable items in crisp and fuzzy environments. Computers & Industrial Engineering, 111, Pp. 272-281.

Haley: The Solid Transportation Problem. 1962.

Hiassat, A., Diabat, A., Rahwan, I., 2017. A genetic algorithm approach for location-inventory-routing problem with perishable products. Journal of manufacturing systems, 42, Pp. 93-103.

Ho, W., Ji, P., 2005. A genetic algorithm for the generalised transportation problem. International journal of computer applications in technology, 22 (4), Pp. 190-197.

Jawahar, N., Balaji, A.N., 2009. A genetic algorithm for the two-stage supply chain distribution problem associated with a fixed charge. European Journal of Operational Research, 194 (2), Pp. 496-537. doi:http://doi.org/10.1016/j.ejor.2007.12.005

Jawahar, N., Gunasekaran, A., Balaji, N., 2012. A simulated annealing algorithm to the multi-period fixed charge distribution problem associated with backorder and inventory. International Journal of Production Research, 50 (9), Pp. 2533-2554.

Kundu, P., Kar, M.B., Kar, S., Pal, T., Maiti, M., 2017. A solid transportation model with product blending and parameters as rough variables. Soft Computing, 21 (9), 2297-2306.

Ojha, A., Das, B., Mondal, S., Maiti, M., 2010. A solid transportation problem for an item with fixed charge, vechicle cost and price discounted varying charge using genetic algorithm. Applied Soft Computing, 10 (1), Pp. 100-110.

Oyewole, G.J., Adetunji, O., 2018. On The Facility Location and FixedCharge Solid Transportation Problem: A Lagrangian Relaxation Heuristic. In: Proceedings of the International Conference on Industrial Engineering and Operations Management, Pretoria/Johnnesburg, South Africa, IEOM Society.

Pérez-Salazar, M.R.R., Mateo-Díaz, N.F., García-Rodríguez, R., Mar-Orozco, C.E., Cruz-Rivero, L., 2015. A genetic algorithm to solve a three-echelon capacitated location problem for a distribution center within a solid waste management system in the northern region of Veracruz, Mexico. Dyna, 82 (191), Pp. 51-57.

Sanei, M., Mahmoodirad, A., Niroomand, S., Jamalian, A., Gelareh, S., 2017. Step fixed-charge solid transportation problem: a Lagrangian relaxation heuristic approach. Computational and Applied Mathematics, 36 (3), Pp. 1217-1237.

Schell, E., 1955. Distributuin of s product by several properties. In: Direstorate of Management Analysis, Proc. of the second Symposium in Linear Programming 2, 615, DCS/Comptroller HQUSAF.

Studio, I.I.C.O., 2016. CPLEX user's manual, Version 12 Release 7. IBM Corp.

Vignaux, G.A., Michalewicz, Z., 1991. A genetic algorithm for the linear transportation problem. IEEE transactions on systems, man, and cybernetics, 21 (2), Pp. 445-452.

Wolsey, L.A., Ceria, S., Cordier, C., Marchand, H., 1998. Cutting planes for integer programs with general integer variables. Mathematical programming, 81 (2), Pp. 201-214.

Yang, L., Liu, L., 2007. Fuzzy fixed charge solid transportation problem and algorithm. Applied soft computing, 7 (3), Pp. 879-889.

Yousefi, K., Afshari, A., Hajiaghaei-Keshteli, M., 2018. Heuristic approaches to solve the fixed-charge transportation problem with discount supposition. Journal of Industrial and Production Engineering, Pp. 1-27.

Zhang, B., Peng, J., Li, S., Chen, L., 2016. Fixed charge solid transportation problem in uncertain environment and its algorithm. Computers & Industrial Engineering, 102, Pp. 186-197.