

[Egyptian Informatics Journal (2012) 13, 123–134](http://dx.doi.org/10.1016/j.eij.2012.05.003)

Cairo University

Egyptian Informatics Journal

[www.elsevier.com/locate/eij](http://www.elsevier.com/locate/eij) [www.sciencedirect.com](http://www.sciencedirect.com/science/journal/11108665)

ORIGINAL ARTICLE

Alternate mutation based artificial immune algorithm for step fixed charge transportation problem

Mahmoud Moustafa El-Sherbiny [\*](#_bookmark0)

*Operations Research Dept., Institute of Statistical Studies and Research (ISSR), Cairo University, Egypt*

Received 21 February 2012; revised 8 May 2012; accepted 20 May 2012

Available online 4 July 2012

Abstract Step fixed charge transportation problem (SFCTP) is considered as a special version of the fixed-charge transportation problem (FCTP). In SFCTP, the fixed cost is incurred for every route that is used in the solution and is proportional to the amount shipped. This cost structure causes the value of the objective function to behave like a step function. Both FCTP and SFCTP are considered to be NP-hard problems. While a lot of research has been carried out concerning FCTP, not much has been done concerning SFCTP. This paper introduces an alternate Mutation based Artificial Immune (MAI) algorithm for solving SFCTPs. The proposed MAI algorithm solves both balanced and unbalanced SFCTP without introducing a dummy supplier or a dummy cus- tomer. In MAI algorithm a coding schema is designed and procedures are developed for decoding such schema and shipping units. MAI algorithm guarantees the feasibility of all the generated solu- tions. Due to the significant role of mutation function on the MAI algorithm’s quality, 16 mutation functions are presented and their performances are compared to select the best one. For this pur- pose, forty problems with different sizes have been generated at random and then a robust calibra- tion is applied using the relative percentage deviation (*RPD*) method. Through two illustrative problems of different sizes the performance of the MAI algorithm has been compared with most recent methods.

© 2012 Faculty of Computers and Information, Cairo University. Production and hosting by Elsevier B.V. All rights reserved.

\* Tel.: +20 01284665997.

KEYWORDS

Fixed charge transportation; Convergence;

Step fixed charge transportation;

Artificial immune algorithm

E-mail address: [m\_sherbiny@yahoo.com](mailto:m_sherbiny@yahoo.com)

1110-8665 © 2012 Faculty of Computers and Information, Cairo University. Production and hosting by Elsevier B.V. All rights reserved.

Peer review under responsibility of Faculty of Computers and Information, Cairo University. <http://dx.doi.org/10.1016/j.eij.2012.05.003>

Production and hosting by Elsevier

1. Introduction

One of the versions of FCTP is the step fixed-charge transpor- tation problem (SFCTP) where the fixed cost is incurred for every route that is used in the solution. In SFCTP, the fixed cost is proportional to the amount shipped. This cost structure causes the value of the objective function to behave like a step function. Considerable work has been done in solving FCTP, such as, lagrangian relaxation method [[1]](#_bookmark27), heuristic approach [[2]](#_bookmark28), genetic algorithm [[3]](#_bookmark29), more-for-less algorithm [[4]](#_bookmark30), branching method [[5]](#_bookmark31), adaptive genetic algorithm [[6]](#_bookmark32) spanning tree-based

genetic algorithm [[7]](#_bookmark20), artificial immune and genetic algorithms based on the spanning tree and Pru¨fer number representation [[8,9]](#_bookmark20). SFCTP was for the first time formulated in 2008 by Kowalski and Lev [[10]](#_bookmark20) and since then has not attracted the attention of researchers. Hence, not much research has been carried out in the area of SFCTP. Balinski in 1961 [[11]](#_bookmark20) has proposed heuristic method for solving FCTP. This method starts with constructing a coefficient matrix and finding the optimal solution based on it. In 1988 Sandrock [[12]](#_bookmark20) analyzed the source induced fixed-charge transportation problem. Since problems with fixed charge are usually NP-hard (Nondetermin- istic Polynomial-time), the computational time to obtain exact solutions increases in a polynomial fashion and very quickly be- comes extremely difficult and long as the size of the problem increases. In the case of the SFCTP due to the step function structure of the objective function *Z* [(1)](#_bookmark1), we are dealing with a ‘‘NP-super hard’’ problem with much ‘‘higher degree’’ of the polynomial complexity [[13,14]](#_bookmark20).

Kowalski and Lev [[10]](#_bookmark20) have followed the approach of Balinski [[11]](#_bookmark20) and have suggested simple heuristic technique based on other two formulae for converting SFCTP to a clas- sical transportation problem. But this heuristic technique is applicable for only small SFCTPs. Altassan et al. [[15]](#_bookmark21) have proposed three more formulae in addition to Balinski’s for- mula [[11]](#_bookmark20) and Kowalski and Lev’s formula [[10]](#_bookmark20) and compared its performance with them.

On the other hand, some special Artificial Immune Systems (AISs) are developed to solve complex optimization and NP- hard problems. One of them is aiNet [[16,17]](#_bookmark22) that is inspired by biological immune system. Opt-aiNet [[16]](#_bookmark22) is an application

Section 3 reviews the previous methods for solving SFCTPs. In Section 4, the main architecture of the proposed MAI algo- rithm and the proposed mutation functions are described, and in Section 5 parametric analysis for the proposed mutation functions (MFs) is carried out. Numerical experiments with proposed MAI algorithm are presented in Section 6. Finally, the conclusion and future work are reported in Section 7.

1. SFCTP description and formulation

Step fixed charge transportation problem (SFCTP) can be described as a distribution problem in which there are *m* sup- pliers (warehouses or factories) and *n* customers (destinations or demand points). Each of the *m* suppliers can ship to any of the *n* customers. Each supplier *i=* 1, 2,..., *m* has *si* units of supply and each customer *j* = 1, 2,.. . , *n* demands *dj* units. *xij* is the unknown quantity to be transported on the route (*i*, *j*) that from supplier *i* to customer *j*. The cost of shipping through route (*i*, *j*) consists of a variable cost equal to the sum- mation of the product of cost per unit *cij* (unit cost for shipping from supplier *i* to customer *j*) and the number of units shipped *xij* plus a fixed cost *fij.* The fixed cost *fij* for route (*i*, *j*) is propor- tional to the transported amount through its route. This con- sists of a fixed cost *fij*,1 for opening the route (*i*, *j*) and an additional cost *fij*,2 when the transported units exceeds a cer- tain amount *Aij.* The objective is to determine which routes are to be opened and the size of the shipment, so that the total cost of meeting demand, given the supply constraints, is mini- mized. The standard mathematical model of SFCTP can be represented as follows [[15]](#_bookmark21):

of aiNet in function optimization. Opt-aiNet considers the *m n*

XX

optimized objective function as antigen, and the candidate

solutions as antibodies. The candidate antibodies evolve according to the matching degree between antibodies and anti- gen that is fitness. The better the matching between them, is the less the mutation degree of candidate antibody. Due to AIS self organizing and learning capability, the AIS has been widely used in many real world complex applications such as job shop scheduling problems [[18,19]](#_bookmark23), multi-objective pro- gramming problems [[20]](#_bookmark25), a hybrid particle swarm [[21]](#_bookmark26) method with artificial immune learning for solving the FCTP [[22]](#_bookmark33), a novel artificial immune algorithm for solving FCTPs [[23]](#_bookmark34), solv-

*Min Z* = (*cijxij* + *bij*;1 *fij*;1 + *bij*;2 *fij*;2) (1)

*i*=1 *j*=1

X

*m*

s.t. *xij* = *dj* for *j* = 1; ... ; *n* (2)

*i*=1

X

*n*

*xij* = *si* for *i* = 1; ... ; *m* (3)

*j*=1

*xij* P 0 ∀*i*; *j*

ing a capacitated FCTP by AI and genetic algorithms with a

Pru¨fer number representation [[8]](#_bookmark20) and student project assign-

ment problem [[24]](#_bookmark35). Also, The AIS algorithms are more effi- cient than the classical heuristic scheduling algorithms such

*bij*;1

= 1 if *xij* > 0

0 otherwise

1 if *xij* > *Aij*

∀*i*; *j*

as such as simulated annealing, tabu search, evolutionary algo-

rithms, and genetic algorithm [[25]](#_bookmark36). While SFCTP is considered as a special version of the FCTP, AIS finds its application in

*bij*;2 =

*m*

0 otherwise

*n*

∀*i*; *j*

solving this complex problem. Therefore AIS is considered

X*si* = X*dj*

one of the feasible approaches for dealing with SFCTPs.

This paper aims to introduce a Mutation based Artificial

*i*=1

*j*=1

Immune (MAI) algorithm for solving SFCTPs and presents

a set of mutation functions in order to improve the quality of the solution. Therefore a set of mutation functions is sug- gested and tested using forty different problems with different dimensions. In addition to that two problems with different sizes are solved to evaluate the performance of the MAI algo- rithm and to compare its performance with the recent five methods.

The rest of the paper is organized as follows: in Section 2, description and mathematical model of SFCTP are presented.

1. Review of methods for solving SFCTPs

As stated earlier, not much work has been done concerning solution of SFCTPs. The existing methods for dealing with SFCTPs are based on using a certain formula for converting the problem into a classical transportation problem and find- ing the solution thereafter.

Balinski [[11]](#_bookmark20) has provided a heuristic solution for FCTP by considering the unit transportation cost of shipping through

the route (*i*, *j*) as in [(4)](#_bookmark3). This method will be denoted as Bal in the remaining part of the paper.

*Cij* = *fij*,1/*Mij* + *cij* (4)

where *Mij* = *Min*(*Si*, *Dj*).

Kowalski and Lev [[10]](#_bookmark20) have proposed two heuristic algo- rithms. In the first algorithm, the formula considered was as in [(5)](#_bookmark3) and in the second algorithm, the formula considered was as in [(6)](#_bookmark3).

*Cij* = (*fij*,1 + *fij*,2)/*Mij* + *cij* (5)

*Cij* = *fij*,2/(*Mij* — *Aij*)+ *cij* (6)

However, the formula [(6)](#_bookmark3) has a few drawbacks [[15]](#_bookmark21). Hence

(5) will be considered as Kow in the remaining part of the paper.

Altassan et al. [[15]](#_bookmark21) have proposed three formulations for *Cij* as in [(7)–(9)](#_bookmark3) and these will be denoted by Alt1, Alt2 and Alt3 respectively in the remaining part of the paper.

code of the main steps for the proposed MAI algorithm for solving SFCTPs is presented as follows:

*C* = *fij*,1 /*Mij* + *cij* if *Aij* P *Mij*

*ij*

(*f*

*ij*,1

+ *fij*,2

)/*M*

*ij*

+ *cij*

if *A*

*ij*

< *Mij*

∀*i*, *j* (7)

*Cij*

= *f*

*ij*,1

/*Mij*

+ *cij*

if *Aij*

P *Mij*

∀*i*, *j* (8)

*Cij*

1. Set number of generations *g* = 1.
2. Apply creating-individual-antibody procedure *PopSize* times to create *PopSize* antibodies *Ai* where *PopSize* represent the population size.
3. Set *i* = 1.
4. Clone *i*th Antibody *Ai* in the population *CN* times.
5. Mutate each of the *CN* clones.
6. Evaluate each of the *CN* clones.
   1. Apply decoding procedure.
   2. Apply Shipping procedure.
   3. Calculate the fitness of each antibody *Ai*.
7. Get the mutated clone with the Best Fitness (*BF)*.
8. If *BF* fitness better than the fitness of *Ai* then replace *Ai* with

*BF.*

1. Set *i* = *i* + 1.
2. Repeat from step 4 to step 9 until *i* > *PopSize* .
3. Calculate the aﬃnity between each two antibodies in the population.
4. Select the antibodies for the new mutation based on the aﬃnity.
5. Create new antibodies to substitute the removed antibodies.

14*. g* = *g* + 1.

15. Repeat step 3 to step 14 until *g* > number of generations.

*fij*,2 /(*Mij* — *Aij* )+ *cij* if *Aij* < *Mij*

= *fij*,1 /*Mij* + *cij* if *Aij* P *Mij fij*,2 /*Aij* + *fij*,1/(*Mij* — *Aij* )+ *cij* if *Aij* < *Mij*

∀*i*, *j*

(9)

The details of the main steps are presented in the following subsections.

In order to improve the local solution of the classical transpor-

tation problem found from converting SFCTP Kowalski and Lev [[19]](#_bookmark24) have proposed a heuristic technique for improving such solution. But such heuristic algorithm can be used for solving only small SFCTPs. This paper introduces an alternate Mutation based Artificial Immune (MAI) algorithm for solv- ing SFCTPs. Further a comparison of the performance and quality of the proposed algorithm is undertaken with the ear- lier proposed methods Bal [[11]](#_bookmark20), Kow [[10]](#_bookmark20), Alt1, Alt2 and Alt3 [[15]](#_bookmark21).

1. The proposed MAI Algorithm

The proposed algorithm in this paper preserves the essential principles of *Opt-aiNet* [[16]](#_bookmark22) algorithm including the cloning, mutation, and clone selection. The implementation of the immune algorithm is often different for each problem handled. That is, the representation and creation of the solutions, the mutation, and the affinity functions should be tailored and implemented to fit the case at hand. In the present paper, the problem of solving the SFCTP has been considered. Altassan et al. [[23]](#_bookmark34) applied artificial immune algorithm (AIA) for solving FCTPs by adding two main procedures for adapting the AIA for solving FCTPs. The first one is the decoding procedure used for splitting the antibody into two orders, one of them to represent the customers’ order and the other to represent the suppliers’ order. The second is the allocating procedure that used for finding the corresponding feasible solution of these orders. In this paper Altassan algorithm [[23]](#_bookmark34) is adapted for solving SFCTPs by replacing the allocation procedure with the proposed shipping procedure which will be used for defin- ing the units *xij* shipped through each route (*i*, *j*). The pseudo

* 1. *Antibody Structure and initialization*

One of the most important issues when designing the AIS lies on its solution (antibody) representation. In order to construct a direct relationship between the problem domain and the MAI, the proposed coding scheme (antibody structure) con- sists of the set of all unrepeated integers in the interval [1, *m* + *n*] in any sequence; where the length of the scheme is equal to *m* + *n*. Therefore, the length of each antibody *Ai* is equal to the sum of the problem dimensions and the suppliers numbers represented by the integer numbers from 1 to *m* and the demands by integer numbers from *m* +1 to *m* + *n*. [Fig. 1](#_bookmark4) depicts a sample antibody which is used to code a 4 · 5 FCTP. As shown in [Fig. 1](#_bookmark4), the cell values are between 1 and 4 + 5. It can be realized that a number is not repeated.

The population is initialized randomly by performing the coding procedure *l* times to create *l* antibodies *Ap* (*p* =1 to *l*), where *l* represents the population size. The Pseudo code for the coding procedure is as follows:

1. Create a collection list *Q* = {1, 2,..., *m* + *n*}.
2. Set *j* = 1.
3. Set *c* = Int(Rand(1, *m* + *n*)) and read the cell *Ap* (*j*)
4. Set *k* = Mod(*c*, Length(*Q*)); where Mod(*c*, Length(*Q*)) is a function that returns the reminder of *c* when it is divided by length(*Q*).
5. Add *Q*[*k*] to the antibody *Ai* in the position *j*.
6. Remove the item *k* from the list *Q*
7. *j* = *j* + 1.
8. Repeat from step 3 to step 7 until *j* > *n* + *m*.
9. Return the antibody *Ap*, where *i* = 1, ..., *l* and *l* is the population size

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 8 | 3 | 9 | 6 | 4 | 7 | 2 | 1 | 5 |

The proposed shipping procedure is used to allocate the transported units *xij* based on the orders (*S* and *D*) resulting from the decoding procedure. In other words, this procedure finds a feasible solution for SFCTP based on the outputs of the decoding procedure. This procedure guarantees the validity of both the first and the second constraints, denoted by [(2) and (3)](#_bookmark2) respectively. Also, this procedure can be used to solve both balanced and unbalanced transportation problems without introducing a dummy supplier or a dummy customer. The Pseudo code for the shipping proce- dure is as follows:

Figure 1 An example of proposed antibody structure.

In this procedure, the Int(Rand(1, *m* + *n*)) is a function that

returns a random integer number in the interval [1, *m* + *n*], Mod(*x*, *y*) is a function that returns the reminder of *x* when it is divided by *y* and *Q*. Remove(*k*) is a function that elimi- nates *k*th element of queue *Q*.

*4.2. Decoding procedure*

This procedure is used to decode the antibody *Ap* into suppli- ers’ order *S* and customers’ order *D*. The inputs of this proce- dure are the generated antibody *Ap*, the number of suppliers *m*, and the number of customers *n* while the results are the sequences of suppliers’ *S* and customers’ *D* [[23]](#_bookmark34). [Fig. 2](#_bookmark5) exhibits the results of applying the decoding procedure on the antibody presented in [Fig. 1](#_bookmark4). The Pseudo code of the decoding proce- dure is illustrated below:

*4.3. Shipping procedure*

1. Set *j* = 1.
2. Read the cell *Ap* (*j*)
3. If *Ap* (*j*) 6 *n* then add *Ap* (*j*) to the supplier order *S*.
4. If *Ap* (*j*)> *n* then add *Ap* (*j*) to the customer order *D.*
5. *j* = *j* + 1.
6. Repeat from step 2 to step 5 until *j* > *n* + *m*.
7. Return the supplier order *S* and the customer order *D.*
8. Set TS = Min (the total Supply, the total demand) and TST = TS.
9. Set L = 1.
   1. Set *i* equal to the L value in suppliers’ order *S* and set *j* equal to the first value in customers’ order *D*. i.e. Set *j* = *D*(1) and *i* = *S*(L).
   2. If *bj* < *ai* and *bj* 6 *Aij*, set *ai* = *ai* — *bj*, *xij* = *bj*,

TS = TS — *bj*, TST = TS — *ai*, remove *D*(1), and L = 0,

* 1. If *bj* < *ai* and *bj* > *Aij*, set *ai* = *ai* — *Aij*, *bj* = *bj* — *Aij*,

*xij* = *Aij*, TS = TS — *Aij*, and TST = TS — *ai*.

TS = TS — *ai*, remove S(L), L = 0, TST = TS, and 2.4. If *bj* = *ai* and *ai* 6 *Aij*, set *xij* = *ai*, *ai* = 0, *bj* = 0, remove *D*(1),

* 1. If *bj* = *ai* and *ai* > *Aij*, set *j* = *D*(1), set *i* = *S*(L), *xij* = *Aij*,

*ai* = *ai* — *Aij*, *bj* = *bj* — *Aij*, TS = TS — *Aij*, and TST = TS — *ai* .

* 1. If *bj* > *ai* and *ai* > *Aij* and (TST — *ai*) P (*bj* — *Aij*),

set *ai* = *ai* — *Aij*, *bj* = *bj* — *Aij*, *xij* = *Aij*, TS = TS — *Aij*, and TST = TS — *ai .*

* 1. If *bj* > *ai* and *ai* 6 *Aij*, set *bj* = *bj* — *ai*, *xij* = *ai*,

TS = TS — *ai*, TST = TS — *ai*, remove *S*(L), and L = L – 1.

* 1. Update L = L + 1

1. Repeat steps 2.1–2.8 until the length of queue *S* <L or the length of queue *D* < 1.
2. If L = the length of queue *S* or the length of queue *D* = 1), set j = *D*(1) and *i* = *S*(L). One of the following states

will occur:

* 1. If *bj* < *ai*, set *ai* = *ai* — *bj*, *xij* = *bj*, TS = TS — *bj*, remove *D*(1), and L = 0.
  2. If *bj* > *ai*, set *bj* = *bj* — *ai*, *xij* = *ai*, TS = TS — *ai*, remove *S*(L), and L = L + 1.
  3. If *bj* = *ai*, set *xij* = *ai*, *ai* = 0, *bj* = 0, TS= TS — *ai*, remove *S*(L), remove *D*(1), and L = 0}

1. Update TST = TS
2. Repeat steps 2–5 until the length of queue *S* plus the length of queue *D* 6 1.
3. Return *xij* 6 *i* = 1, 2,..., *m* and *j* = 1, 2,.. . , *n*.

# *Input*:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 8 | 3 | 9 | 6 | 4 | 7 | 2 | 1 | 5 |

*Ap*

*j*=1

# *Output:*

*D* The customers' order

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 4 | 5 | 2 | 3 | 1 |

*S* The Suppliers' order

|  |  |  |  |
| --- | --- | --- | --- |
| 3 | 4 | 2 | 1 |

Figure 2 Illustrative example of the decoding procedure.

The inputs of shipping procedure are the sequence of sup- pliers *S* and the sequence of customers *D* (the output of decod- ing procedure). Based on these orders, the shipping procedure allocates units *xij* (feasible solution) of FCTP. [Fig. 3](#_bookmark6) exhibits the results of applying the shipping procedure on the result presented in [Fig. 2](#_bookmark5).

* 1. *Evaluation of the solutions*

As mentioned above each antibody is decoded and the result is used as an input for shipping procedure. The solution resulted from shipping procedure is evaluated using objective function *Z*, as denoted in (1). The value of objective function is assigned to the antibody as its fitness.

* 1. *Cloning and mutation*

Each antibody is cloned CN number of times, where CN de- notes the Cloning Number. The clones are then mutated to get new antibodies that are different from their parents. In the proposed MAI algorithm, four basic Mutation Functions (MFs) together with other twelve hybrid MFs are proposed as explained below:

The final allocation is:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ***D1*** | | ***D2*** | ***D3*** | ***D4*** | ***D5*** | ***Si*** | ***TSi*** |
| ***S1*** |  | 5 | 15 |  |  | 20 |  |
| ***S2*** |  | 15 | 15 |  |  | 60 | 30 |
| ***S3*** |  |  |  | 15 | 15 | 30 |  |
| ***S4*** |  | 10 |  | 15 | 15 | 40 |  |
| ***dj*** | 30 | 30 | 30 | 30 | 30 | **150** |  |
| ***TDj*** | 30 |  |  |  | | **30** |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ***D1*** | | ***D2*** | ***D3*** | ***D4*** | ***D5*** | ***Si*** | ***TSi*** |
| ***S1*** | | | |  |  | 20 | 20 |
| ***S2*** | | | |  |  | 60 | 60 |
| ***S3*** | | | | 15 | 15 | 30 |  |
| ***S4*** | | | |  | 15 | 40 | 25 |
| ***dj*** | 30 | 30 | 30 | 30 | 30 | **150** |  |
| ***TDj*** | 30 | 30 | 30 | 15 |  | **105** |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | ***D1*** | ***D2*** | ***D3*** | ***D4*** | ***D5*** | ***Si*** | ***TSi*** |
| ***S1*** |  | 5 | 15 |  |  | 20 |  |
| ***S2*** | 30 | 15 | 15 |  |  | 60 |
| ***S3*** |  |  |  | 15 | 15 | 30 |
| ***S4*** |  | 10 |  | 15 | 15 | 40 |
| ***dj*** | 30 | 30 | 30 | 30 | 30 | **150** |  |
| ***TDj*** |  |  | | | |  |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ***D1*** | | ***D2*** | ***D3*** | ***D4*** | ***D5*** | ***Si*** | ***TSi*** |
| ***S1*** | | | |  |  | 20 | 20 |
| ***S2*** | | | |  |  | 60 | 60 |
| ***S3*** | | | | 15 | 15 | 30 |  |
| ***S4*** | | | | 15 | 15 | 40 | 10 |
| ***dj*** | 30 | 30 | 30 | 30 | 30 | **150** |  |
| ***TDj*** | 30 | 30 | 30 |  |  | **90** |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ***D1*** | | ***D2*** | ***D3 D4*** | ***D5*** | ***Si*** | ***TSi*** |
| ***S1*** | |  |  |  | 20 | 20 |
| ***S2*** | |  |  |  | 60 | 60 |
| ***S3*** | |  | 15 | 15 | 30 |  |
| ***S4*** | | 10 | 15 | 15 | 40 |  |
| ***dj*** | 30 | 30 | 30 30 | 30 | **150** |  |
| ***TDj*** | 30 | 20 | 30 | | **80** |  |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ***dj si***  ***xij*** | 5  3  15 | 5  4  15 | 4  3  15 | 4  4  15 | 2  4  10 | 2  2  15 | 2  1  5 | 3  2  15 | 3  1  15 | 1  2  30 |

Figure 3 An illustrative example of applying the shipping algorithm.

The *first basic MF* is the two point swap (2PointSwap) MF and it is based on generating two random numbers *j* and *k*

where *j*, *k* 2 [1, *n* + *m*], i.e. *j* and *k* = Int(Rand(1, *n* + *m*)) [[21,22]](#_bookmark26). Therefore swap the two antibody digits corresponding

to these two random numbers. The 2PointSwap MF is pre- sented in [Fig. 4](#_bookmark7).

*The second basic MF* is based on generating two random numbers *j*, *k* where *j* = Int(Rand(1, *n*)) and *k* = Rand(1, *n* + *m*) and inverse the order of the antibody’s digits between these two random numbers (*j*, *k*) [[21,22]](#_bookmark26). The inverse swap

*Ai*

*j k*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 5 | 2 | 7 | 4 | 3 | 1 | 6 |

Muted *Ai*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 5 | 2 | 1 | 3 | 4 | 7 | 6 |

Figure 5 Inversion swap mutation function *InvSwap* (*j*, *k*).

*Ai*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 5 | 2 | 7 | 4 | 3 | 1 | 6 |

(*InvSwap*) MF is presented in [Fig. 5](#_bookmark8).

Muted *Ai*

*j j+1*

*Ai*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ***D1*** | | ***D2*** | ***D3*** | ***D4*** | ***D5*** | ***Si*** | ***TSi*** |
| ***S1*** | |  |  |  |  | 20 | 20 |
| ***S2*** | | 15 |  |  |  | 60 | 45 |
| ***S3*** | |  |  | 15 | 15 | 30 |  |
| ***S4*** | | 10 |  | 15 | 15 | 40 |  |
| ***dj*** | 30 | 30 | 30 | 30 |  | **150** |  |
| ***TDj*** | 30 | 5 | 30 | | | **65** |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ***D1*** | | ***D2*** | ***D3*** | ***D4*** | ***D5*** | ***Si*** | ***TSi*** |
| ***S1*** | | 5 |  |  |  | 20 | 15 |
| ***S2*** | | 15 |  |  |  | 60 | 45 |
| ***S3*** | |  |  | 15 | 15 | 30 |  |
| ***S4*** | | 10 |  | 15 | 15 | 40 |  |
| ***dj*** | 30 | 30 | 30 | 30 | 30 | **150** |  |
| ***TDj*** | 30 |  | 30 | | | **60** |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ***D1*** | | ***D2*** | ***D3*** | ***D4*** | ***D5*** | ***Si*** | ***TSi*** |
| ***S1*** | | | | |  | 20 | 20 |
| ***S2*** | | | | |  | 60 | 60 |
| ***S3*** | | | | | 15 | 30 | 15 |
| ***S4*** | | | | |  | 40 | 40 |
| ***dj*** | **30** | **30** | **30** | **30** | **30** | **150** |  |
| ***TDj*** | 30 | 30 | 30 | 30 | 15 | **135** |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ***D1*** | | ***D2*** | ***D3*** | ***D4*** | ***D5*** | ***Si*** | ***TSi*** |
| ***S1*** |  | 5 |  |  |  | 20 | 15 |
| ***S2*** |  | 15 | 15 |  |  | 60 | 30 |
| ***S3*** |  |  |  | 15 | 15 | 30 |  |
| ***S4*** |  | 10 |  | 15 | 15 | 40 |  |
| ***dj*** | 30 | 30 | 30 | 30 | 30 | **150** |  |
| ***TDj*** | 30 |  | 15 |  | | **45** |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ***D1*** | | ***D2*** | ***D3*** | ***D4*** | ***D5*** | ***Si*** | ***TSi*** |
| ***S1*** | | | | |  | 20 | 20 |
| ***S2*** | | | | |  | 60 | 60 |
| ***S3*** | | | | | 15 | 30 | 15 |
| ***S4*** | | | | | 15 | 40 | 25 |
| ***dj*** | 30 | 30 | 30 | 30 | 30 | **150** |  |
| ***TDj*** | 30 | 30 | 30 | 30 |  | **120** |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 5 | 2 | 7 | 3 | 4 | 1 | 6 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 5 | 2 | 7 | 4 | 3 | 1 | 6 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| *D* | 5 | 4 | 2 | 3 | 1 | The Customers order |
| *S* | 3 | 4 | 2 | 1 | The | Suppliers order |
|  |  |  |  |  |  |  |

*j k*

Muted *Ai*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 5 | 2 | 1 | 4 | 3 | 7 | 6 |

Figure 4 Two point swap mutation *2PointSwap* (*j*, *k*).

Figure 6 Neighbor swap mutation *NeibSwap* (*j*, *j* + 1).

The *NeibSwap* based on generating a random numbers *j* 2 *The third basic MF* is the Neighbor swap (*NeibSwap*) MF.

[1, *n* + *m*] and swap the positions *j* and *j* + 1. I.e. Generate

*j* = Int(Rand (1, *n* + *m*)) and swap the positions *j* and *j* + 1.

*NS* = *MNS*(1—(1—*NF*)*u* ) (13)

The neighbor swap MF is presented in [Fig. 6](#_bookmark9).

The *fourth basic MF* is a uniform random number where a

*NF* = *Lowest Fitness* — *Fitness*

*Lowest Fitness* — *Highest Fitness*

(14)

fixed number of swaps is setup for all antibodies during all iter-

ations. This fixed number is donated by *MNS*. The number of swaps (*NS*) for this *MF* is represented by [(10)](#_bookmark10), and is fixed dur- ing all iterations.

*NS* = *MNS* (10)

The next three MFs are based on the 2PointSwap MF [[24]](#_bookmark35),

followed by two MFs each based on *InvSwap MF, NeibSwap.* The other five *MFs* proposed are functions of two parameters. The first parameter is the non-uniform factor based on which the number of swaps is determined. The second parameter is the degree of non-uniformity (*u*). These MFs are designed to

*The 13th MF* is designed to be inversely related with the ratio

(*T*) of the current iteration number (*CIN*) and the total number of iterations (*TNI*). That is, the more the search goes on; the less is the number of swaps. This is really intuitive as in con- trast to the first stages of the search where a real exploration of the search space through significant changes in the solutions are required, at the last stages of the search fine tuning with little changes of the supposed-to-be near-optimal solutions is more reasonable [[23]](#_bookmark34). The number of swaps (*NS*) for this mutation is represented in [(15)](#_bookmark12) where *u* is the degree of non- uniformity.

be directly related with *u*.

*The fifth MF* is based on generating a random number

*NS* = *MNS*(1—*Tu* ), where *T* = *CIN*

*TNI*

(15)

*NS* ∈ [1, *n* + *m*]. Therefore the *2PointSwap* is performed *NS*

times. The number of swaps (*NS*) for this mutation is repre-

sented in [(11)](#_bookmark11)

*NS* = Int(Rand(1, *n* + *m*)) (11)

*The sixth MF* is based on a uniform random number located in

the range of 10–30% of the sum of problem dimensions (*n* + *m*). The number of swaps (*NS*) for this mutation is repre- sented by [(12)](#_bookmark14), where *r* is a random number in the interval [0.1, 0.3].

*NS* = Int(Rand(1, *r*(*m* + *n*))) (12)

*The seventh MF* is based on time where more is the time

elapsed; less will be the number of swaps. First start with applying random number of two-points-swap till a pre-defined ratio of time is elapsed. After that the two points swap MF is applied for the remaining time. The time is represented by the ratio of current iteration to the total number of iterations.

swap times or *InvSwap* MFs. A random number *r* ∈ [0, 1] is *The eighth MF* is based on applying either non-uniform generated and if *r* > pre-defined value *v*, then the non-uniform

swap time will be applied; else *InvSwap* MF will be applied.

*The ninth MF* is based on the time where more is the time elapsed; less will be the number of swaps. First start with applying random number of *swap* till the time passes a pre-de- fined ratio. After that the *InvSwap MF* is applied for the remaining time.

*r* ∈ [1, (*n* + *m*)/2] and repeating *NeibSwap r* times for each *The 10th MF* is based on generating a random numbers antibody *AP*.

*The 11th MF* is based on the time where more is the time elapsed; less will be the number of swaps. First start with applying random number of *NeibSwap* MF till a pre-defined ratio of time elapsed. After that the *NeibSwap* MF is applied for the remaining time.

*The twelfth MF* is based on the fitness of the solution [[24]](#_bookmark35). As the SFCTP is a minimization problem, the function is de- signed to be directly related with the Normalized Fitness (*NF*) of the solution. That is, solutions with normalized fitness closer to one, i.e. relatively bad solutions, will be subject to more number of swaps. This actually gives the chance for low affinity solutions to mutate more in order to improve their affinities. The *NS* for this MF is adopted as [(13)](#_bookmark12) and the nor- malized fitness of each antibody is calculated using [(14)](#_bookmark12).

*The 14th MF* is based on both the time and the normalized fit- ness of the solution. It basically uses the average of these two factors to decide the number of swaps. Basically, the MF is de- signed to be directly related with the fitness but inversely re- lated to the time [[24]](#_bookmark35). The average of time (T) and normalized Fitness (*TF*) is calculated as represented in [(16)](#_bookmark13) and the number of swaps for this mutation is adopted as [(17)](#_bookmark15).

1

*TF* = 2 (*NF* + (1 — *T*)) (16)

*NS* = *MNS*(1—(1—*TF*)*u* ) (17)

In the *15th* and the *16th MFs*, a random factor (*R*) is included

so that the number of swaps is based on the non-uniform fac- tor, time and fitness respectively, but with some randomiza- tion. The random factor *R* takes values between zero and one [[23]](#_bookmark34). The functions behave almost the same way as the ori- ginal ones when Rand is close to zero. The closer the *R* to one is; the closer is the number of swaps to the max swaps number. These two MFs allow the search to escape from local optima by occasionally increasing the number of swaps. The numbers of swaps for these mutations are adopted as in [(18) and](#_bookmark16) [(19)](#_bookmark16), respectively.

*NS* = *MNS* × *R*(1—*NF*)*u* (18)

*NS* = *MNS* × *R*(*Tu* ) (19)

* 1. *Affinity function*

The calculations of the affinity *AF* (similarity) between each pair of antibodies are applied to prevent similar solutions with high evaluation from being copied to the next generation and hence dominating the search. The selection of the antibodies from one generation to the next depends on calculation of the affinity among all the antibodies of the current generation. This is tech- nically applied to reduce the chance of a premature convergence to local optima. The technique used to check the similarity be- tween every two antibodies in a population is through counting the number of similar digits in the two solutions. The basic idea is that the more the number of similar variables in the two anti- bodies is, the higher the similarity between them. Based on a spe- cific parameter, the algorithm eliminates those solutions that

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Table 1 Characteristics of SFCT test problems.  Problem Range of supply and Rang of variable Rang of fixed costs Rang of fixed costs Rang of step values size demand costs (*cij*) (*fij*,1) (*fij*,2) (*Aij*) | | | | | | | | | | | | | | |
|  | Lower limit | Upper limit |  | Lower limit | Upper limit |  | Lower limit | Upper limit |  | Lower limit | Upper limit |  | Lower limit | Upper limit |
| 3 · 3 | 50 | 100 |  | 1 | 3 |  | 10 | 20 |  | 20 | 50 |  | 50 | 100 |
| 4 · 5 | 150 | 250 |  | 1 | 9 |  | 10 | 40 |  | 30 | 70 |  | 150 | 250 |
| 5 · 10 | 200 | 500 |  | 1 | 9 |  | 10 | 50 |  | 30 | 90 |  | 200 | 500 |
| 10 · 10 | 300 | 500 |  | 1 | 9 |  | 100 | 200 |  | 200 | 400 |  | 300 | 500 |
| 10 · 15 | 500 | 1000 |  | 1 | 9 |  | 100 | 500 |  | 200 | 600 |  | 500 | 1000 |
| 15 · 15 | 500 | 2000 |  | 1 | 9 |  | 100 | 500 |  | 200 | 600 |  | 500 | 2000 |
| 15 · 20 | 1000 | 3000 |  | 1 | 9 |  | 100 | 500 |  | 200 | 700 |  | 1000 | 3000 |
| 20 · 20 | 1000 | 3000 |  | 1 | 9 |  | 100 | 500 |  | 200 | 700 |  | 1000 | 3000 |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

have *AF* more than a specific parameter -Number of Similarities (*NS*). The affinity function of two antibodies *Ap* and *Ak* is repre- sented as in [(20)](#_bookmark17).

*AF*(*A* , *A* )= X*y*

*p*

*k*

*i*

*i*

obtained best solution, respectively. After converting the objective values to *RPD*s, the mean *RPD* is calculated for each trial. Eight problems with different size are generated using a designed Microsoft Excel spreadsheet and used to discover

the best MF from the proposed 16 mutation functions. The

where *y* =

1 if the *i*th gene of *Ap*

= the *i*th gene of *Ak*

characteristics of these problems are used in [[23]](#_bookmark34) as FCTPs

and adopted for presenting SFCTPs by adding additional costs

*i* 0 Otherwise

*fij*,2

and step values *Aij*. The characteristics of these problems

1. Parametric analysis

(20)

are presented in Table 1.

All the 40 problems considered were solved to find the total cost of the associated SFCTP and subsequently the corre- sponding *RPD*s for each of the proposed 16 MFs. The values of average *RPD*s, based on five illustrative examples for each

Due to the important affect of the mutation functions in the performance of the artificial immune algorithm, in this section a calibration of the proposed MAI algorithm through discov- ering the best *MF* from the implemented 16 functions is pre- sented. Because the scale of the objective functions in each problem is different, they could not be compared directly. Therefore, the Relative Percentage Deviation (*RPD*) is used for each combination [[26]](#_bookmark37). *RPD* is calculated by using [(21)](#_bookmark18).

*RPD* = *A*lg*sol* — *Minsol* × 100 (21)

*Minsol*

where *Algsol* and *Minsol* are the obtained objective values for each replication of trial in a given combination and the

Table 2 The comparative results of the average *RPD* for the proposed mutation functions.

Mutation function *Average RPD* of the test problems Overall mean *RPD* Rank of MFs

of the eight dimensions considered using the six methods and the overall mean *RPD* for each of the methods are presented in Table 2.

Based on the results presented in Table 2 and [Fig.](#_bookmark19) 7, the overall mean *RPD* of the proposed AMIA algorithm with the 13th MF is providing the least value as compared to other mutation functions. This is followed by the 11th and sixth MFs. Further, the overall mean *RPD* of the proposed AMIA algorithm with the first, 10th, third and fourth MFs is provid- ing the largest values in that order while it is providing a mod- erate values with remaining MFs. The ranking based on the performance for all MFs is illustrated in Table 2. Hence, it can be concluded that the proposed MAI algorithm with the

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 5 · 10 | 10 · 10 | 10 · 15 | 15 · 15 | 15 · 20 20 · 20 | |  | |
| 1 | 3.8 | 3.7 | 11.4 | 12.1 | 13.8 | 25.5 | 14.0 | 16 |
| 2 | 0.5 | 2.5 | 7.5 | 10.3 | 7.7 | 17.7 | 9.4 | 10 |
| 3 | 3.1 | 1.5 | 7.6 | 10.9 | 12.6 | 22.4 | 12.0 | 14 |
| 4 | 0.7 | 3.4 | 6.3 | 10.1 | 12.6 | 22.3 | 11.7 | 13 |
| 5 | 0.3 | 2.0 | 3.4 | 7.9 | 10.4 | 17.1 | 8.9 | 9 |
| 6 | 0.4 | 1.7 | 4.9 | 5.5 | 8.0 | 10.4 | 6.5 | 3 |
| 7 | 1.3 | 1.8 | 4.9 | 7.2 | 8.7 | 16.1 | 8.3 | 7 |
| 8 | 2.0 | 2.9 | 7.0 | 8.5 | 13.1 | 19.1 | 11.0 | 12 |
| 9 | 0.3 | 2.0 | 4.0 | 7.1 | 9.9 | 16.0 | 8.5 | 8 |
| 10 | 0.5 | 1.9 | 9.2 | 11.1 | 12.2 | 24.0 | 12.4 | 15 |
| *11* | *1.3* | *0.4* | *5.2* | *5.1* | *2.3* | *11.6* | *5.0* | *2* |
| 12 | 0.3 | 0.5 | 5.6 | 7.0 | 9.6 | 11.5 | 7.5 | 4 |
| 13 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1 |
| 14 | 0.5 | 1.2 | 4.0 | 7.7 | 7.6 | 14.8 | 7.6 | 5 |
| 15 | 0.1 | 7.0 | 5.6 | 9.5 | 12.2 | 17.4 | 10.6 | 11 |
| 16 | 0.7 | 0.4 | 5.4 | 6.6 | 9.7 | 15.2 | 8.2 | 6 |

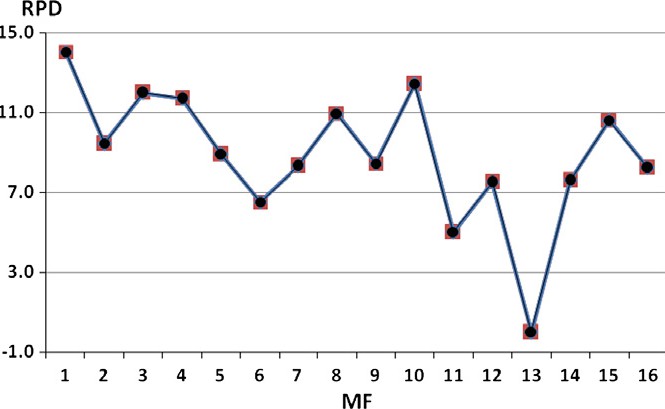


Figure 7 Fitted mean plot for *RPD* at each MF.

13th MF is superior and can be used as the best alternative for finding a local solution for SFCTPs as compared to other MFs.

In addition to the above, in order to statistically test the significance of effectiveness of the results using different meth- ods, the paired sample *t*-tests were used to determine the signif- icant differences in the *RPD* values obtained using the 16 MFs, for each of the pairs. For the purpose of comparisons the *RPD* values obtained using all the 40 problems were used. The results of the tests are summarized in Table 3.

As illustrated in Table 3, it can be concluded at 0.01 level of significance the quality of the results using the 11th MF is very close to the 13th one and both are superior to the others. But the 13th MF is most superior. This corroborates the results obtained based on the *RPD* analysis. Therefore, in the next section, the 13th MF will be used with the MAI algorithm in comparing with the Bal [[11]](#_bookmark20), Kow [[10]](#_bookmark20), Alt1, Alt2 and Alt3

[[15]](#_bookmark21) methods for solving SFCTPs.

1. Numerical experiments

In order to prove the superiority of the proposed MAI algo- rithm, in the following subsections, two illustrative examples

are presented in order to bring out the differences between the proposed MAI algorithm with the erstwhile result of using 13th MF and the above mentioned methods.

* 1. *Illustrative problems*

To illustrate the performance of the proposed MAI algorithm, two problems with different sizes, previously addressed by Altassan et al. [[15]](#_bookmark21) are solved and compared with the solutions provided by Balinski [[11]](#_bookmark20), Kowalski and Lev [[10]](#_bookmark20) and Altassan et al. [[15]](#_bookmark21), for solving SFCTPs. The sizes of the problems are 4 · 5 and 5 · 10 respectively. The coefficient matrix of each problem is generated using each method and this matrix is solved by using the classical transportation module in the QM package for Windows Version 2.1 (QM is a package for quantitative methods, operations research and management science) for finding the corresponding local solution of each method. Subsequently, the corresponding costs of each local solution are calculated using designed Microsoft Excel spread- sheets for this purpose. The data, the parameters, the gener- ated coefficient matrix, the local solution, and the cost items for each problem using each method are presented in the fol- lowing paragraphs, in addition to the results found by the proposed MAI algorithm.

Concerning the first problem, the problem size is considered to be 4 · 5 with variable costs, and the fixed costs for the prob- lem as given in Table 4. The coefficient matrices generated using the Bal, Kow, Alt1, Alt2 and Alt3 methods and their cor- responding local solutions using QM are presented in Tables 5 and 6 respectively. While the coefficient matrices of the Alt1 and Alt2 are different as illustrated in Table 5 their local opti- mal distribution are the same as illustrated in Table 6. The corresponding local solution using the proposed MAI algo- rithm is presented in Table 7.

While the step values *Aij* for all *i* and *j* are equal to 20 as illus- trated in Table 4, the optimal distributions have exceed such values in different cells (*xij* units) as illustrated in bold font in Table 6. As per the SFCTP model, only when the shipped units *xij* exceeds *Aij*, the additional opening cost *fij*,2 is applied. It can be observed that the optimal distributions using the Bal, Kow, Alt1, Alt2, and Alt3 methods have exceeded the step value in 3,

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Table 3 The *p*-values of paired sample *t*-tests of the mutation functions.  MF *p*-Value (2-tailed) | | | | | | | | | | | | | | | |
|  | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 |
| 1 | 0.004 | 0.018 | 0.003 | 0.000 | 0.000 | 0.000 | 0.029 | 0.002 | 0.012 | 0.003 | 0.006 | 0.003 | 0.004 | 0.027 | 0.013 |
| 2 | 0.030 | 0.473 | 0.002 | 0.000 | 0.096 | 0.008 | 0.050 | 0.025 | 0.352 | 0.029 | 0.006 | 0.176 | 0.173 | 0.046 |  |
| 3 | 0.005 | 0.146 | 0.004 | 0.000 | 0.000 | 0.001 | 0.463 | 0.003 | 0.089 | 0.003 | 0.008 | 0.003 | 0.596 |  |  |
| 4 | 0.024 | 0.267 | 0.017 | 0.000 | 0.004 | 0.003 | 0.194 | 0.016 | 0.336 | 0.017 | 0.022 | 0.027 |  |  |  |
| 5 | 0.280 | 0.057 | 0.170 | 0.000 | 0.315 | 0.080 | 0.007 | 0.274 | 0.023 | 0.372 | 0.055 |  |  |  |  |
| 6 | 0.049 | 0.016 | 0.166 | 0.002 | 0.615 | 0.576 | 0.011 | 0.066 | 0.016 | 0.071 |  |  |  |  |  |
| 7 | 0.739 | 0.029 | 0.292 | 0.000 | 0.513 | 0.073 | 0.004 | 0.991 | 0.011 |  |  |  |  |  |  |
| 8 | 0.020 | 0.714 | 0.016 | 0.000 | 0.002 | 0.005 | 0.050 | 0.015 |  |  |  |  |  |  |  |
| 9 | 0.697 | 0.021 | 0.272 | 0.000 | 0.518 | 0.107 | 0.005 |  |  |  |  |  |  |  |  |
| 10 | 0.010 | 0.114 | 0.006 | 0.000 | 0.001 | 0.001 |  |  |  |  |  |  |  |  |  |
| 11 | 0.149 | 0.008 | 0.172 | 0.000 | 0.168 |  |  |  |  |  |  |  |  |  |  |
| 12 | 0.639 | 0.011 | 0.882 | 0.000 |  |  |  |  |  |  |  |  |  |  |  |
| 13 | 0.000 | 0.000 | 0.000 |  |  |  |  |  |  |  |  |  |  |  |  |
| 14 | 0.348 | 0.020 |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 15 | 0.029 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 10 | 5 | 3 | 2 | 4 | 6 |  | 40 | 20 | 30 | 20 | 10 |  | 50 | 70 | 80 | 70 | 80 |  | 20 | 20 | 20 | 20 | 20 |
| 100 | 3 | 5 | 3 | 4 | 3 |  | 10 | 20 | 20 | 30 | 20 |  | 60 | 70 | 60 | 80 | 60 |  | 20 | 20 | 20 | 20 | 20 |
| 20 | 3 | 4 | 6 | 5 | 2 |  | 40 | 30 | 10 | 20 | 30 |  | 60 | 80 | 80 | 70 | 70 |  | 20 | 20 | 20 | 20 | 20 |
| 70 | 2 | 5 | 4 | 3 | 4 |  | 10 | 40 | 40 | 10 | 10 |  | 80 | 40 | 50 | 50 | 50 |  | 20 | 20 | 20 | 20 | 20 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Table 5 The coefficient matrices of the first problem using the different methods.  Method *Si D*1 *D*2 *D*3 *D*4 *D*5  Bal *S*1 9.0 5.0 5.0 6.0 7.0  *S*2 3.3 6.0 3.3 7.0 3.3  *S*3 5.0 5.5 6.5 7.0 3.5  *S*4 2.3 7.0 4.6 4.0 4.2 | | | | | | |
| Kow | *S*1 | 14.0 | 12.0 | 13.0 | 13.0 | 15.0 |
|  | *S*2 | 4.8 | 9.5 | 4.1 | 15.0 | 4.3 |
|  | *S*3 | 8.0 | 9.5 | 10.5 | 14.0 | 7.0 |
|  | *S*4 | 4.3 | 9.0 | 5.3 | 9.0 | 5.0 |
| Alt1 | *S*1 | 9.0 | 5.0 | 5.0 | 6.0 | 7.0 |
|  | *S*2 | 4.8 | 6.0 | 4.1 | 7.0 | 4.3 |
|  | *S*3 | 5.0 | 5.5 | 6.5 | 7.0 | 3.5 |
|  | *S*4 | 4.3 | 7.0 | 5.3 | 4.0 | 5.0 |
| Alt2 | *S*1 | 9.0 | 5.0 | 5.0 | 6.0 | 7.0 |
|  | *S*2 | 6.0 | 6.0 | 4.2 | 7.0 | 4.5 |
|  | *S*3 | 5.0 | 5.5 | 6.5 | 7.0 | 3.5 |
|  | *S*4 | 6.0 | 7.0 | 5.0 | 4.0 | 5.3 |
| Alt3 | *S*1 | 9.0 | 5.0 | 5.0 | 6.0 | 7.0 |
|  | *S*2 | 6.5 | 6.0 | 6.4 | 7.0 | 6.5 |
|  | *S*3 | 5.0 | 5.5 | 6.5 | 7.0 | 3.5 |
|  | *S*4 | 6.5 | 7.0 | 7.3 | 4.0 | 6.8 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Table 6 Optimal distributions of the first problem using the different methods.  Method *Si D*1 *D*2 *D*3 *D*4 *D*5  Bal *S*1 10  *S*2 70 30  *S*3 10 10  *S*4 40 10 20 | | | | | | |
| Kow | *S*1 |  | 0 |  | 10 |  |
|  | *S*2 |  |  | 70 |  | 30 |
|  | *S*3 |  | 20 |  |  | 0 |
|  | *S*4 | 40 |  |  |  | 30 |
| Alt1 | *S*1 |  | 10 |  |  |  |
|  | *S*2 |  | 10 | 70 |  | 20 |
|  | *S*3 |  |  |  |  | 20 |
|  | *S*4 | 40 |  |  | 10 | 20 |
| Alt2 | *S*1 |  | 10 |  |  |  |
|  | *S*2 |  | 10 | 70 |  | 20 |
|  | *S*3 |  |  |  |  | 20 |
|  | *S*4 | 40 |  |  | 10 | 20 |
| Alt3 | *S*1 |  |  | 10 |  |  |
|  | *S*2 |  | 20 | 60 |  | 20 |
|  | *S*3 |  |  |  |  | 20 |
|  | *S*4 | 40 |  |  | 10 | 20 |

4, 2, 2, and 2 cells respectively while MAI algorithm exceeds the step value in only one cell (Tables 6 and 7). This observation

Table 7 Optimal distribution of the first problem using the

MAI algorithm.

*D*1 *D*2

*S*1

*S*2 *S*3 *S*4

20

*D*3

10

60

*D*4

*D*5

0

20

20

10

20

20

20

illustrates the total of the additional cost P*i*=1P*j*=1 *fij*,2 using

*m*

*n*

MAI algorithm has the smallest total additional cost compared to other methods (see Table 8)*.*

The comparative study of the total costs for the first prob- lem using the different methods is summarized in Table 8. It can be observed that the proposed MAI algorithm provides the best solution with least total cost among the all methods, while Alt1 and Alt2 methods have the second best solution fol- lowed by Alt3.

Concerning the second problem, the problem size is consid- ered to be 5 · 10 with variable costs, and the fixed costs for the problem as given in Table 9. The coefficient matrices generated using the designed Excel spreadsheet based on the Bal, Kow, Alt1, Alt2 and Alt3 methods and presented in Table 10. The cor- responding coefficient matrices are solved using QM package and presented in Table 11. The corresponding local solution using the proposed MAI algorithm is presented in Table 12.

Table 8 Summary of total costs of the first problem using the

different methods.

Method

P P

*m n*

*i*=1 *j*=1

*fij*,1

P P

*m n*

*i*=1 *j*=1

*fij*,2

P P

*m n*

*i*=1 *j*=1

*c x* Total cost

*ij ij*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Bal | 150 | 200 | 580 | 930 |
| Kow | 110 | 250 | 620 | 980 |
| Alt1 | 140 | 140 | 580 | 860 |
| Alt2 | 140 | 140 | 580 | 860 |
| Alt3 | 150 | 140 | 590 | 880 |
| MAI | 180 | 60 | 610 | 850 |

As illustrated in the SFCTP model, *fij*,2 is applied only when the shipped units *xij* exceeds *Aij* . Hence in Table 4 the optimal

distributions (*xij*

Table 4 The parameters and variables of the first problem (size 4 · 5).

*Si*

1

*Dj*

40

2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5

20

70

10

60

Variable cost *cij* Fixed cost *fij*,1 Fixed cost *fij*,2 Step value *Aij*

units) have exceeded *Aij*

in different cells as

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *si* | 1 | 2 | 3 | 4 | | 5 | 6 | 7 | 8 | 9 | | 10 | 1 | | 2 | 3 | 4 | 5 | 6 | 7 | | 8 | 9 | 10 |
|  | *dj* |  |  |  | |  |  |  |  |  | |  |  | |  |  |  |  |  |  | |  |  |  |
|  | 40 | 20 | 50 | 10 | | 10 | 20 | 30 | 30 | 50 | | 40 |  | |  |  |  |  |  |  | |  |  |  |
| Variable cost *cij Aij* | | | | | | | | | | | | | | | | | | | | | | | | |
| 20 | 4 | 5 | 5 | | 2 | 2 | 4 | 4 | 2 | | 8 | 4 |  | 40 | 30 | 40 | 50 | 40 | 30 | | 20 | 40 | 50 | 40 |
| 40 | 4 | 4 | 7 | | 5 | 6 | 5 | 7 | 6 | | 7 | 5 |  | 10 | 50 | 30 | 40 | 30 | 50 | | 20 | 30 | 20 | 10 |
| 90 | 4 | 6 | 3 | | 8 | 4 | 3 | 3 | 3 | | 5 | 7 |  | 50 | 40 | 40 | 10 | 50 | 20 | | 30 | 10 | 30 | 20 |
| 60 | 5 | 6 | 3 | | 6 | 6 | 4 | 6 | 8 | | 2 | 2 |  | 40 | 10 | 30 | 20 | 20 | 40 | | 50 | 20 | 20 | 30 |
| 90 | 3 | 5 | 5 | | 8 | 3 | 8 | 5 | 7 | | 4 | 6 |  | 20 | 30 | 20 | 20 | 10 | 30 | | 50 | 20 | 40 | 50 |
| Fixed cost *fij*,1 Fixed cost *fij*,2 | | | | | | | | | | | | | | | | | | | | | | | | |
| 100 | | 170 | 190 | | 100 | 170 | 150 | 190 | 170 | | 150 | 200 | 210 | | 400 | 280 | 370 | 320 | 210 | | 300 | 220 | 230 | 210 |
| 110 | | 170 | 170 | | 200 | 180 | 160 | 180 | 180 | | 170 | 140 | 290 | | 340 | 340 | 280 | 360 | 330 | | 200 | 390 | 310 | 400 |
| 120 | | 120 | 170 | | 100 | 120 | 170 | 130 | 160 | | 110 | 190 | 360 | | 300 | 330 | 290 | 290 | 400 | | 310 | 210 | 350 | 390 |
| 130 | | 120 | 130 | | 180 | 160 | 140 | 170 | 180 | | 190 | 110 | 390 | | 220 | 220 | 250 | 330 | 290 | | 370 | 310 | 350 | 280 |
| 110 | | 180 | 160 | | 170 | 130 | 120 | 110 | 160 | | 160 | 120 | 340 | | 320 | 270 | 270 | 270 | 320 | | 360 | 220 | 370 | 280 |

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Table 10 The coefficient matrices for the second problem using different methods. | | | | | | | | | | | |
| Method | *Si* | *D*1 | *D*2 | *D*3 | *D*4 | *D*5 | *D*6 | *D*7 | *D*8 | *D*9 | *D*10 |
| Bal | *S*1 | 9.0 | 13.5 | 14.5 | 12.0 | 19.0 | 11.5 | 13.5 | 10.5 | 15.5 | 14.0 |
|  | *S*2 | 6.8 | 12.5 | 11.3 | 25.0 | 24.0 | 13.0 | 13.0 | 12.0 | 11.3 | 8.5 |
|  | *S*3 | 7.0 | 12.0 | 6.4 | 18.0 | 16.0 | 11.5 | 7.3 | 8.3 | 7.2 | 11.8 |
|  | *S*4 | 8.3 | 12.0 | 5.6 | 24.0 | 22.0 | 11.0 | 11.7 | 14.0 | 5.8 | 4.8 |
|  | *S*5 | 5.8 | 14.0 | 8.2 | 25.0 | 16.0 | 14.0 | 8.7 | 12.3 | 7.2 | 9.0 |
| Kow | *S*1 | 19.5 | 33.5 | 28.5 | 49.0 | 51.0 | 22.0 | 28.5 | 21.5 | 27.0 | 24.5 |
|  | *S*2 | 14.0 | 29.5 | 19.8 | 53.0 | 60.0 | 29.5 | 19.7 | 25.0 | 19.0 | 18.5 |
|  | *S*3 | 16.0 | 27.0 | 13.0 | 47.0 | 45.0 | 31.5 | 17.7 | 15.3 | 14.2 | 21.5 |
|  | *S*4 | 18.0 | 23.0 | 10.0 | 49.0 | 55.0 | 25.5 | 24.0 | 24.3 | 12.8 | 11.8 |
|  | *S*5 | 14.3 | 30.0 | 13.6 | 52.0 | 43.0 | 30.0 | 20.7 | 19.7 | 14.6 | 16.0 |
| Alt1 | *S*1 | 9.0 | 13.5 | 14.5 | 12.0 | 19.0 | 11.5 | 13.5 | 10.5 | 15.5 | 14.0 |
|  | *S*2 | 14.0 | 12.5 | 19.8 | 25.0 | 24.0 | 13.0 | 19.7 | 12.0 | 19.0 | 18.5 |
|  | *S*3 | 7.0 | 12.0 | 13.0 | 18.0 | 16.0 | 11.5 | 7.3 | 15.3 | 14.2 | 21.5 |
|  | *S*4 | 8.3 | 23.0 | 10.0 | 24.0 | 22.0 | 11.0 | 11.7 | 24.3 | 12.8 | 11.8 |
|  | *S*5 | 14.3 | 14.0 | 13.6 | 25.0 | 16.0 | 14.0 | 8.7 | 19.7 | 14.6 | 9.0 |
| Alt2 | *S*1 | 9.0 | 13.5 | 14.5 | 12.0 | 19.0 | 11.5 | 13.5 | 10.5 | 15.5 | 14.0 |
|  | *S*2 | 13.7 | 12.5 | 41.0 | 25.0 | 24.0 | 13.0 | 27.0 | 12.0 | 22.5 | 18.3 |
|  | *S*3 | 7.0 | 12.0 | 36.0 | 18.0 | 16.0 | 11.5 | 7.3 | 13.5 | 22.5 | 26.5 |
|  | *S*4 | 8.3 | 28.0 | 14.0 | 24.0 | 22.0 | 11.0 | 11.7 | 39.0 | 13.7 | 30.0 |
|  | *S*5 | 20.0 | 14.0 | 14.0 | 25.0 | 16.0 | 14.0 | 8.7 | 29.0 | 41.0 | 9.0 |
| Alt3 | *S*1 | 9.0 | 13.5 | 14.5 | 12.0 | 19.0 | 11.5 | 13.5 | 10.5 | 15.5 | 14.0 |
|  | *S*2 | 36.7 | 12.5 | 35.3 | 25.0 | 24.0 | 13.0 | 35.0 | 12.0 | 31.0 | 49.7 |
|  | *S*3 | 7.0 | 12.0 | 28.3 | 18.0 | 16.0 | 11.5 | 7.3 | 32.0 | 22.2 | 36.0 |
|  | *S*4 | 8.3 | 40.0 | 16.8 | 24.0 | 22.0 | 11.0 | 11.7 | 41.5 | 25.8 | 22.3 |
|  | *S*5 | 25.5 | 14.0 | 23.8 | 25.0 | 16.0 | 14.0 | 8.7 | 34.0 | 29.3 | 9.0 |
|  |  |  |  |  |  |  |  |  |  |  |  |

shown in bold font in Table 11. It can be observed that the opti- mal distributions using Bal, Kow, Alt1, Alt2, and Alt3 methods have exceeded the step value in 3, 4, 1, 2, and 1 cells (*xij* units) respectively while MAI algorithm did not exceed it in any cell (see Tables 11 and 12). This illustrates the reason for total of

Table 9 The parameters and variables of the second problem (size 5 · 10).

the additional cost P P *fij* 2 being positive for each meth-

*m n*

,

*i*=1

*j*=1

ods and while the same equal to zero for MAI (see Table 13)*.* In addition to that, the comparative study of the total costs for the second problem using different methods is summarized in Table 13. It can be observed that the proposed MAI algo- rithm provides the best solution with least total cost among

all methods, while Alt1 method has the second best solution followed by Alt3.

From the above two illustrated problems and based on the results summarized in Tables 8 and 13, it can be observed that the proposed MAI algorithm provides the best solution as compared to the earlier proposed methods. Therefore, it can be concluded that the solution quality of the proposed MAI algorithm is superior to the rest.

In order to further explore the effectiveness of the proposed MAI algorithm, the results based on different problems with eight dimensions ranging from 3 · 3 to 20 · 20 and with

Table 12 Optimal distribution of the second problem using

the MAI algorithm.

*D*1 *D*2 *D*3 *D*4

10

*D*5

10

*D*6 *D*7

*D*8

0

30

*D*9 *D*10

*S*1

*S*2 *S*3 *S*4 *S*5

10

40

20

20

10

30

20

0

30

30

40

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *S*1  *S*2 | 20 | 10 |  | 0 | 10  20 |  |  | Method | P*m* P*n fij*,1 | P*m* P*n fij*,2 | P*m* P*n cijxij* | Total cost |
| *S*3 | 40 |  |  | 20 30 | |  |  | Bal | 1790 | 1200 | 960 | 3950 |
| *S*4 |  | 50 |  | 0 |  | 10 |  | Kow | 1500 | 1640 | 1140 | 4280 |
| *S*5 |  |  | 10 |  |  | 40 | 40 | Alt1 | 1770 | 220 | 1150 | *3140* |

different *Aij* were analyzed. The details of analysis and the re- sults are presented in the next section.

* 1. *Comparative study*

The aim of this section is to prove whether the solutions pro- vided by the proposed MAI algorithm are significantly better than solutions provided by other methods. This is accom- plished by using RPDs for ranking the methods and statisti- cally comparing the significance of results using the paired sample *t*-tests. The characteristics of the forty problems with eight different dimensions, as illustrated in Table 1 are used. All the 40 problems considered were solved to find the total cost of the associated SFCTP and subsequently the corre- sponding *RPD*s for each of the earlier proposed methods (Bal, Bow, Alt1, Alt2, and Alt3) and the proposed MAI algo- rithm. The values of average *RPD*s, based on five illustrative examples for each of the eight dimensions considered using

Table 14 The comparative results of the average *RPD* for the proposed methods.

Method *Average RPD* of the test problems

Overall mean *RPD*

the six methods and the overall mean *RPD* for each of the methods are presented in Table 14.

Based on the results presented in Table 14, the overall mean *RPD* of the proposed MAI algorithm is providing the least value as compared to the other methods. This is followed by the Alt1 method. Hence, it can be concluded that the proposed MAI algorithm is superior and can be used as the best alterna- tive for finding a local solution for SFCTPs as compared to the earlier used methods.

Table 11 Optimal distributions of the second problem using

different methods.

Method

Bal

*Si*

*S*1 *S*2 *S*3 *S*4 *S*5

*S*1 *S*2 *S*3 *S*4 *S*5

*D*1 *D*2 *D*3 *D*4 *D*5 *D*6 *D*7 *D*8 *D*9 *D*10

10

10

20

10

10

30

20

30

30

30

10

0 40

50

Kow

0

20

40

20

0

30

10

0

30

30

20

40

10

50

Alt1

Table 13 Summary of total costs for the second problem

using the different methods.

*i*=1 *j*=1

*i*=1 *j*=1

*i*=1 *j*=1

Alt2 1770

Alt3 1810

MAI 1780

620

220

0

1190

1460

1220

3580

3490

3000

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Alt2 | *S*1 |  |  |  | 10 |  |  |  | 10 |  |  |
|  | *S*2 |  | 20 |  |  |  |  |  | 20 |  |
|  | *S*3 | 40 | 0 |  |  | 10 | 10 | 30 |  |  |
|  | *S*4 |  |  | 0 |  |  | 10 |  |  | 50 |
|  | *S*5 |  |  | 50 |  |  |  |  |  |  | 40 |
| Al3 | *S*1 |  |  |  |  |  |  |  |  | 20 |  |
|  | *S*2 |  | 10 |  |  |  |  |  | 30 |  |  |
| *S*3 | | 40 | 0 |  | 10 |  | 10 |  | 30 | |  |
| *S*4 | |  |  | 50 |  |  | 10 |  |  | |  |
| *S*5 | |  | 10 |  |  | 10 |  | 30 |  | | 40 |

In addition to the above, in order to statistically test the sig- nificance of effectiveness of the results using different methods, the paired sample *t*-tests were used to determine the significant differences in the *RPD* values obtained using the six methods, for each of the pairs. For the purpose of comparisons the *RPD* values obtained using all the 40 problems were used. The re- sults of the paired sample *t*-tests are summarized in Table 15.

As illustrated in Table 15, it can be concluded at 0.01 level of significance the quality of the results using the proposed MAI algorithm is the best, followed by the Alt1 method [[15]](#_bookmark21) considering the total cost which is significantly lower than those provided by the rest of the methods. Hence, the pro- posed MAI algorithm can be considered as the best alternative as compared to the other methods (Bal, Kow, Alt1, Alt2 and Alt3) provided by Balinski [[11]](#_bookmark20), Kowalski and Lev [[10]](#_bookmark20) and Altassan et al. [[15]](#_bookmark21) for solving SFCTPs respectively.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | 3 · 3 | 4 · 5 | 5 · 10 | 10 · 10 | 10 · 15 | 15 · 15 | 15 · 20 | 20 · 20 |  |  |
| MAI | *0.0* | *0.0* | *0.0* | *0.0* | *0.0* | *0.0* | *0.0* | *0.0* | *0.0* |
| Bal | 2.1 | 2.8 | 10.5 | 10.2 | 9.1 | 2.3 | 2.4 | 14.5 | 6.7 |  |
| Bow | 4.6 | 4.4 | 17.0 | 16.2 | 16.3 | 7.9 | 11.8 | 21.6 | 12.5 |  |
| Alt1 | 0.0 | 4.0 | 4.1 | 7.0 | 7.0 | 1.2 | 10.3 | 4.6 | 4.8 |  |
| Alt2 | 0.5 | 4.6 | 11.7 | 18.8 | 15.7 | 1.7 | 13.6 | 12.7 | 9.9 |  |
| Alt3 | 3.6 | 2.0 | 13.9 | 11.0 | 16.0 | 3.0 | 20.7 | 8.5 | 9.8 |  |
|  |  |  |  |  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Bal | Bow | Alt1 | Alt2 | Alt3 |
| MAI | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| Alt3 | 0.412 | 0.217 | 0.002 | 0.840 |  |
| Alt2 | 0.385 | 0.142 | 0.000 |  |  |
| Alt1 | 0.000 | 0.000 |  |  |  |
| Bow | 0.993 |  |  |  |  |

1. Conclusion

This paper has proposed an alternate Mutation based Artifi- cial Immune (MAI) algorithm for solving SFCTPs. The MAI algorithm solves both balanced and unbalanced SFCTP without introducing a dummy supplier or a dummy customer. Although MAI algorithm with population-based search is characterized as an evolutionary-like algorithm, the major contributions are the coding schema and the decoding proce- dure that avoid infeasibility of any candidate solutions. All the generated antibodies are feasible solutions for SFCTP. Be- sides, due to the significant role of mutation function on the MAI algorithm’s quality, 16 different mutation functions are implemented and its performances are compared using RPD for selecting the best one. Also, the comparative study of the MAI algorithm with the method proposed by Balinski [[11]](#_bookmark20), Kowalski and Lev [[10]](#_bookmark20) and Altassan et al. [[15]](#_bookmark21) for solving SFCTPs showed that the MAI algorithm is superior to the others. The performance of MAI algorithm and the solution quality prove that MAI algorithm is highly competitive and can be considered as a viable alternative to solve SFCTPs.

Future work includes Investigating using other metaheuris- tic techniques combined with the proposed decoding and ship- ping algorithms for solving other problems such as capacitated FCTP, Multi-Step FCTP. In addition, it is proposed to carry out further experimentation with parameters of the MAI algo- rithm and testing the proposed MAI algorithm on other real life problems.

References

1. Wright D, Haehling von Lanzenauer C. Solving the fixed charge problem with lagrangian relaxation and cost allocation heuristics. Eur J Oper Res 1989;42:304–12.
2. Wright D, Haehling von Lanzenauer C. COLE: a new heuristic approach for solving the fixed charge problem Computational results. Eur J Oper Res 1991;52:235–46.
3. Su S, Zhan DC. New genetic algorithm for the fixed charge transportation problem. In: Proceedings of the 6th world congress on, intelligent control and automation; 2006. p. 7039–43.
4. Adlakha V, Kowalski K, Vemuganti R R, Lev Benjamin. More- for-less algorithm for fixed-charge transportation problems, OMEGA. Int J Manage Sci 2007;35(1):116–27.
5. Adlakha V, Kowalski K, Lev Benjamin. A branching method for the fixed charged transportation problem, OMEGA. Int J Manage Sci 2010;38(5):393–7.
6. Othman Zalinda, Delavar Mohammad-Reza Rostamian, Behnam Sarah, Lessanibahri Sina. Adaptive genetic algorithm for fixed- charge transportation problem. In: International multi conference

of engineers & computer sciences (IMECS), March; 2011. p. 16– 18.

1. Hajiaghaei, Keshteli M, Molla-Alizadeh-Zavardehi S, Tavakkoli- Moghaddam R. Addressing a nonlinear fixed-charge transporta- tion problem using a spanning tree-based genetic algorithm. Comput Ind Eng 2010;59:259–71.

Table 15 The *p*-values of paired sample *t*-tests proposed methods.

Method *p*-Value (2-tailed)

1. Molla-Alizadeh-Zavardehi S, Hajiaghaei-Keshteli M, Tavakkoli- moghaddam R. Solving a capacitated fixed-charge transportation problem by artificial immune and genetic algorithms with a Pru¨fer number representation. Expert Syst Appl 2011;38:10462–74.
2. El-Sherbiny Mahmoud M. Comments on ‘‘Solving a capacitated fixed-charge transportation problem by artificial immune and genetic algorithms with a Pru¨fer number representation’’ by Molla-Alizadeh-Zavardehi, S. et al. Expert Systems with Appli- cations (2011). Expert Syst Appl 2012;39:11321–22.
3. Kowalski K, Lev Benjamin. On step fixed-charge transportation problem OMEGA. Int J Manage Sci 2008;36(5):913–7.
4. Balinski ML. Fixed cost transportation problem. Naval Res Log Quart 1961;8:41–54.
5. Sandrock K. A simple algorithm for solving small, fixed-charge transportation problem. J Oper Res Soc 1988;39(5):467–75.
6. Kowalski K, Lev Benjamin. New approach to fixed charges problems (FCP). Int J Manage Sci Eng Manage 2007;2(1):75–80.
7. Lev Benjamin, Kowalski K. Modeling fixed-charge problems with polynomials OMEGA. Int J Manage Sci 2011;39(6):725–9.
8. Altassan Khaled M. El-Sherbiny Mahmoud M, Bokkasam S. In: Handling the step fixed charge transportation problem (ICDeM2012), Kedah, Malaysia, March 15–18, 2012.
9. Timmis J, Knight T, Catro Hart LN. An overview of artificial immune systems. Computation in cells and tissues: perspectives and tools thought. Nat Comput Ser. Springer-Verlag; 2004, p. 51– 86.
10. De Castro LN, Timmis J. An artificial immune network for multimodal function optimization. In: Proc. of IEEE congress on, evolutionary computation, vol. 1; 2002. p. 699–704.
11. Engin Orhan, Do¨yen Alper. A new approach to solve hybrid flow shop scheduling problems by artificial immune system. Future Gener Comput Syst 2004;20:1083–95.
12. Tsai JT, Ho WH, Liu TK, Chou JH. Improved immune algorithm for global numerical optimization and job shop scheduling problems. Appl Math Comput 2007;194:406–24.
13. Abd El-Wahed Waiel F, Zaki Elsayed M, El-Refaey Adel M. Artificial immune system based neural networks for solving multi- objective programming problems. Egypt Inform J 2010; 11(2):59–65.
14. El-Sherbiny Mahmoud M. Particle swarm inspired optimization algorithm without velocity equation. Egypt Inform J 2011; 12(1):1–8.
15. El-Sherbiny M. Mahmoud, Alhamali M. Rachid. A hybrid particle swarm method with artificial immune learning for solving the fixed charge transportation problem. Comput Ind Eng 1: 1–2, submitted for publication.
16. Altassan Khalid M, El-Sherbiny Mahmoud M, Abid Ahmed D. A novel artificial immune algorithm for solving fixed charge trans- portation problems. In: International conference on innovation and information management (ICIIM2012), Chengdu, China, January 7–8, 2012.
17. El-Sherbiny Mahmoud M, Ibrahim Yasser M. An artificial immune algorithm with alternative mutation methods: applied to the student project assignment problem. In: International conference on innovation and information management (ICI- IM2012), Chengdu, China, January 7–8, 2012.
18. Hart E, Ross P. Immune system approach to scheduling in changing environments. In: Proceedings of the genetic and evolutionary computation conference 1999 (GECCO’99), Florida, USA, July 13–17, 1999. p. 1559–66.
19. Taguchi G. Introduction to quality engineering. White Plains: Asian Productivity Organization/UNIPUB; 1986.