A genetic algorithm with local search for solving single-source single-sink nonlinear non-convex minimum cost flow problems

Ghasemishabankareh, Behrooz; Ozlen, Melih; Li, Xiaodong; Deb, Kalyanmoy

Ghasemishabankareh, B., Ozlen, M., Li, X., & Deb, K. (2020). A genetic algorithm with local search for solving single-source single-sink nonlinear non-convex minimum cost flow problems. Soft Computing, 24, 1153–1169. https://doi.org/10.1007/s00500-019-03951-2
Document Version: Accepted Manuscript

Published Version: https://doi.org/10.1007/s00500-019-03951-2

Soft Computing
(https://doi.org/10.1007/s00500-019-03951-2)

A Genetic Algorithm with Local Search for Solving Single-Source Single-Sink Nonlinear Non-Convex Minimum Cost Flow Problems

Behrooz Ghasemishabankareh · Melih Ozlen · Xiaodong Li · Kalyanmoy Deb

Published online : 20 March 2019

Abstract Network models are widely used for solving difficult real-world problems. The minimum cost flow problem (MCFP) is one of the fundamental network optimisation problems with many practical applications. The difficulty of MCFP depends heavily on the shape of its cost function. A common approach to tackle MCFPs is to relax the non-convex, mixed-integer, nonlinear program (MINLP) by introducing linearity or convexity to its cost function as an approximation to the original problem. However, this sort of simplification is often unable to sufficiently capture the characteristics of the original problem. How to handle MCFPs with non-convex and nonlinear cost functions is one of the most challenging issues. Considering that mathematical approaches (or solvers) are often sensitive to the shape of the cost function of non-convex MINLPs, this paper proposes a hybrid genetic algorithm (GA) with local search (namely GALS) for solving single-source single-sink nonlinear non-convex MCFPs. Our experimental results demonstrate that GALS offers highly competitive performances as compared to those of the mathematical solvers and a standard genetic algorithm.

Keywords Minimum cost flow problem · Non-convex cost function · Genetic algorithm · Local search

B. Ghasemishabankareh · M. Ozlen · X. Li
School of Science, RMIT University, Melbourne, Australia
E-mail: behrooz.ghasemishabankareh@rmit.edu.au
E-mail: melih.ozlen@rmit.edu.au
E-mail: xiaodong.li@rmit.edu.au

K. Deb
Department of Electrical and Computer Engineering, Michigan State University, East Lansing, MI 48824 USA
E-mail: kdeb@egr.msu.edu

1 Introduction

Network models are widely used in practice for solving difficult real-world problems, including a wide range of network optimisation problems such as the shortest path problem, the assignment problem, the maximum flow problem, the minimum cost flow problem (MCFP), the spanning tree problem, etc. Among these, MCFP is one of the fundamental network optimisation problems with many practical applications, e.g., supply chain, logistics, transportation, and production planning [1]. Since the shortest path problem and the maximum cost flow problem are special cases of MCFPs, in this study we consider MCFP as a generic type of network flow models.

The complexity of MCFP highly depends on the shape of its chosen cost function, which computes how much commodities can be sent through the network. An MCFP with a linear cost function is polynomial solvable [1, 2]. However, the linear cost function may not be able to adequately express the actual cost in a practical situation [3]. For instance, in cargo transportation, factors such as the amount of transportation and transport distance affect the transportation cost function. As a result, the transportation cost may decrease while the amount of cargo increases due to the economy of scale [3]. This scenario shows that the linear and convex cost functions may not be adequate in modelling the real-world scenarios of MCFPs. In contrast, nonlinear nonconvex cost functions that do not make this assumption should represent better the real-world characteristics of the network flow problems [4, 5].

A large-scale non-convex MCFP is NP-hard and often considered a challenging problem to be solved in a short period of time, since there are numerous extreme points in a solution set [6, 7]. MCFP using a concave cost function is known to be NP-hard [7], where complexity arises from the fact that minimising the concave cost function over a convex feasible region does not guarantee that the global optimum will be found [8].

A single-source uncapacitated MCFP is a special type of MCFPs which has been studied in the past decade. Both exact and approximation methods exist for solving concave MCFPs, among which the branch-and-bound technique is one of the most popular methods for solving single-source uncapacitated MCFPs. For instance, constraint programming and linear programming methods were hybridised with the branch-and-bound method to solve fix-charged MCFP [9]. This hybrid method is twice faster than a commercial integer programming codes. A branch-and-cut algorithm is proposed to solve a single commodity uncapacitated MCFP [10] which consisted of the Steiner tree problem, uncapacitated lot-sizing problems, and the fixed-charge transportation problems as special cases. Other branch-and-bound methods for solving the MCFP can be found in the following works [11–13].

Dynamic programming is another popular mathematical approach for solving MCFPs. Fontes et al. [15] proposed a dynamic programming algorithm to optimally solve single-source uncapacitated MCFPs with linear and concave cost functions, whereas Burkard et al. [14] applied a dynamic programming approach to solve a single-source uncapacitated MCFP using a linear approximation of the concave cost function. Furthermore, Erickson et al. [16] proposed a dynamic programming approach called send-and-split method to solve concave MCFPs with single-source and uncapacitated arcs. Kovacs [17] presented an extensive survey on various mathematical programming techniques applied for solving MCFPs.

In addition, many attempts have been made to solve concave single-source uncapacitated MCFPs using metaheuristic methods such as ant colony optimisation (ACO) and GA. Monteiro et al. [8] proposed a hybrid method combining ACO and local search to solve single-source uncapacitated MCFPs. They also carried out a sensitivity study on the parameters used in the ACO algorithm. Other ACO based algorithms for solving the same problems were presented in [3, 18]. A hybrid method combining GA with local search was also proposed to solve single-source uncapacitated MCFPs and the results were compared with the dynamic programming approach and the upper bound obtained by a local search method [19]. All the aforementioned methods were able to solve the uncapacitated network instances, with the largest network instance considered being networks with 50 nodes.

Literature review suggests only a few limited studies can be found on network flow optimisation using nonlinear non-convex cost functions [5, 20, 21] mostly focusing on small-sized problems. For example, a nonlinear non-convex transportation problem was solved using an GA [22], and by two exact methods [23, 24].

In this paper, we propose a hybrid GA with local search (namely GALS) method to solve the nonlinear non-convex single-source single-sink MCFP. A key novelty of our proposed method is that we use GA to evolve the representation scheme and then local search to refine the searching capability of GALS. Since many real-world MCFPs consist of large-sized networks, this paper shows that GALS is able to handle large-scale MCFPs more effectively. We evaluate our proposed method on a set of 45 network instances, and compare the results with that of a state-of-the-art mathematical solve package, as well as a standard GA. Our results suggest the superiority of GALS over the mathematical solver and the standard GA in solving large-scale MCFPs.

The rest of the paper is structured as follows: the problem definition is presented in Section 2 and the proposed GALS is described in Section 3. Section 4 presents the problem instances and experimental results. Finally the conclusion and future research directions are provided in Section 5.

2 Problem definition

Let $G(\mathcal{N}, A)$ be a directed network with $\mathcal{N} = 1, \ldots, n$ nodes and a set of m directed arcs A. The upper and lower bounds of flow on each arc (i,j) are denoted by $u_{i,j}$ and $l_{i,j}$ respectively. Instead of a linear or convex cost function, a nonlinear non-convex cost function f is considered in this paper and assigned to each arc [1]. q represents the supply/demand in the source/sink nodes respectively. x_{ij} is an integer number that denotes the amount of flow through an arc (i, j). Generally in MCFP, we aim to send flows throughout the network to satisfy all demands by minimising the total cost (i.e., the objective function value). Fig. 1 provides a single-source single-sink MCFP example with $n = 5$ nodes and $m = 7$ arcs. In this example, the aim is to find a flow which satisfies all demands in node 5 (sink) by sending all supplies from node 1 (source) in order to minimise the total cost through the network. The total cost of the single-source single-sink MCFP can be minimised according to the following [1]:

$$Minimise : Z(\mathbf{x}) = \sum_{i=1}^{n}\sum_{j=1}^{n} f(x_{ij}), \tag{1}$$

$$\text{s.t.} \quad \sum_{j=1}^{n} x_{ij} - \sum_{k=1}^{n} x_{ki} = \begin{cases} q, & \text{if } i = 1 \\ 0, & \text{if } i = (2, 3, \dots, n-1) \\ -q, & \text{if } i = n \end{cases} \tag{2}$$

$$l_{ij} \le x_{ij} \le u_{ij}, \quad (i, j = 1, \dots, n), \tag{3}$$

$$x_{ij} \in \mathbb{Z}, \quad (i, j = 1, \dots, n), \tag{4}$$

where the cost function in Eq. (1) minimises the total cost within the network. Eq. (2) is the flow balance constraint in which the difference between the first term (total outflow) and the second term (total inflow) is equal to q and $-q$ for source and sink nodes respectively, and is equal to 0 otherwise. Eq. (3) states that the flow on each arc should be within the lower and upper bounds, and finally Eq. (4) ensures that all flow values are integer. Several assumptions of the network we have here include: 1) The network is directed; 2) the network does not contain two or more arcs with the same tail and head nodes; 3) the supply and demand for all nodes except source and sink nodes are equal to 0; 4) the lower bound for each arc (l_{ij}) has a value of 0; 5) there are no negative cycles in the network; 6) the cost function on each arc ($f(x_{ij})$) is a nonlinear non-convex function, rather than a linear or convex one.

3 The proposed GA method

In this paper, we will demonstrate how a hybrid GA with local search can be applied for solving the MCFP. GA is a stochastic search algorithm inspired by natural selection and genetics [25]. Basically, GA has five main components: representation, a process to create new solutions, evaluation of fitness, genetic operators and parameters [26]. Obviously *representation* plays a key role in solving many optimization problems (e.g., MCFP), before the GA search can be carried out. In the following section, we will first describe issues associated with the commonly-used priority-based representation scheme, and then we present the proposed GALS for solving MCFPs.

[1] Some example formulations of nonlinear non-convex cost functions for MCFPs are presented in Section 4.

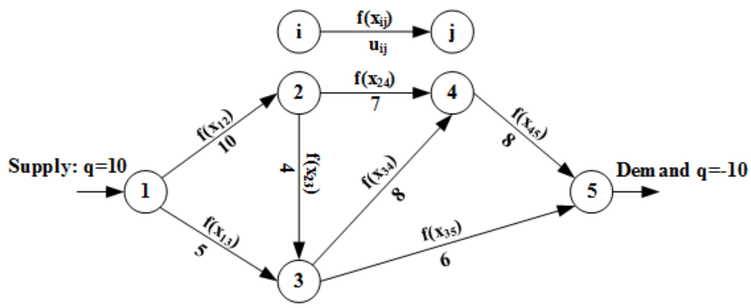


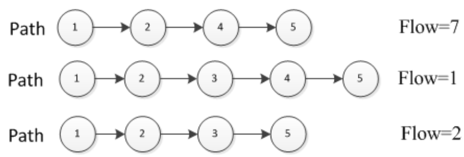
Fig. 1: A single-source single-sink MCFP example.

3.1 Issues with priority-based representation

Priority-based representation has been widely adopted in solving project scheduling, shortest path and network design problems [27–29]. It is one of the most popular approaches to represent an MCFP [30]. In a priority-based representation scheme for MCFPs, the number of genes is equal to the number of nodes (n), and the *allele* (i.e., the possible value each gene can take) is created randomly between 1 and the number of nodes (n) (Fig. 2a).

Nodes:	1	2	3	4	5
Priorities	5	4	1	3	2

(a) A chromosome for the priority-based encoding method.

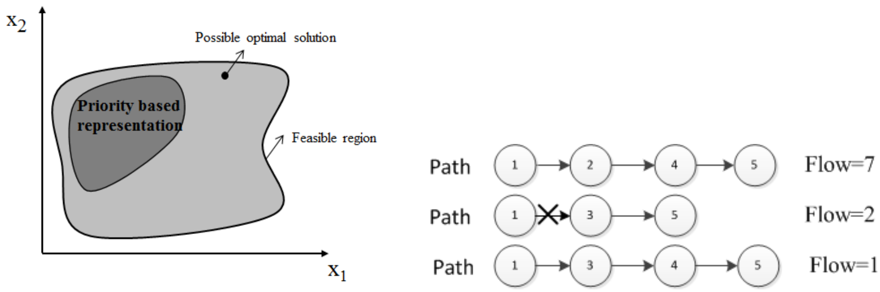


(b) Paths produced from the priority chromosome shown in (a).

Fig. 2: A chromosome and its MCFP solution for the network in Fig. 1.

From the priority chromosome, several paths can be constructed in order to satisfy the demand of MCFP presented in Fig. 1. These paths constitutes an appropriate MCFP solution. As shown in Fig. 2b, we can construct several paths starting from node 1, ending in node 5. For each path, starting from node 1, we select the successor node with a higher priority. For example, the successors for node 1 are nodes 2 and 3. Based on the priority chromosome, the priorities of nodes 2 and 3 are values of 4 and 1 respectively. Hence node 2 is chosen. After node 2, the successor nodes are nodes 3 and 4. Since the priority of node 3 (1) is smaller than that of node 4 (3), node 4 is chosen as the destination and from node 4 the only possible successor node is node 5. Finally, the completed path is $1 \rightarrow 2 \rightarrow 4 \rightarrow 5$. The possible flow on this path is equal to the minimum of capacities on the arcs of the path and the supply/demand (Flow=min{10,7,8,q=10}=7). At this point, we need to update the capacities on the arcs, supply, and demand. If the demand in node 5 is not satisfied, then the second path is constructed, similarly as the previous path. This process continues until the demand is satisfied. Fig. 2b shows the 3 paths produced from the above process.

The priority-based representation method is unable to encode all the possible solutions in the feasible search space, as shown in Fig. 3a. For example, Fig. 3b shows an MCFP instance, where the priority-based representation method fails to realise.



(a) The priority-based encoding only represents partially the feasible space.

(b) A feasible MCFP solution that priority-based encoding fails to represent.

Fig. 3: Disadvantages of the priority-based encoding method.

3.2 Improved priority-based encoding

To address the above issue, we propose an improved priority-based encoding (iPE) method here. In iPE, the *locus*, (i.e., the position (or index) of the gene) and *allele* of the main chromosome are identical to the priority-based encoding method. The main difference is that after the first path is constructed, from the second path onwards, instead of using the same chromosome, two genes of the main chromosome are randomly swapped and the new path is then constructed based on this new chromosome. In other words, it is now possible to generate MCFP solutions based on newly produced representation instances, rather than one fixed presentation instance. Algorithm 1 shows the procedure of iPE in detail. By using the swapping technique in the main chromosome, the priority-based method is redeemed and it is now possible to represent more feasible solutions using this iPE method.

3.3 GALS for solving single-source single-sink MCFPs

Building on iPE method, this section proposes GALS for solving large-scale MCFPs with nonlinear non-convex cost functions, where iPE is used to per-

Algorithm 1 iPE decoding procedure

procedure Input($Priority, Supply, Demand, Network$)
  Path=1
  while $supply \neq 0$ and $Demand \neq 0$ **do**
    if Path=1 **then**
      $Chromosome_{Path} \leftarrow$ Main chromosome
    else
      $Chromosome_{Path} \leftarrow$ Swap two genes randomly in the main chromosome.
    end if
    Construct a path:
    Construct a path according to the $Chromosome_{Path}$, following the priority-based encoding procedure.
    Send a feasible flow:
    Check the maximum capacity of all arcs in the path ($MaxC$)
    Send an integer flow $min\{MaxC, Supply\}$
    update: Supply, Demand and Network
    Path$\leftarrow$Path+1
  end while
end procedure

form local search to further enhance the searching capability of GALS, which involves the following procedure:

Initialisation: A population of n_{pop} individuals (chromosomes) is first randomly generated (according to Subsection 3.1).

Crossover and mutation: Crossover and mutation operators are then applied to create a new offspring population. For each newly generated offspring, two parents are first randomly selected and a weight mapping crossover (WMX) is applied [30]. Subsequently an inversion mutation operator is applied [30].

Solution decoding: To counteract the limitations of the priority-based representation, the iPE decoding procedure (as shown in Algorithm 1) is performed N times for each chromosome in the population. As shown in Fig. 4, after performing the decoding, N solutions are obtained.

Evaluation and local search: The N number of MCFP solutions generated from the previous decoding step (with respect to each chromosome) are evaluated using Eq. (1). A local search is carried out by selecting the decoded solution with the smallest cost, among all N solutions.

Population update: After evaluating all individuals in the population, the tournament selection procedure (with a tournament size of 2) is applied to select the fitter individuals for the next generation.

Termination criteria: The above process continues until a stopping criterion is met, which is either 1) no further fitness value improvement in the best individual of the population for β successive iterations; or 2) the maximum number of function evaluations (NFEs) is reached.

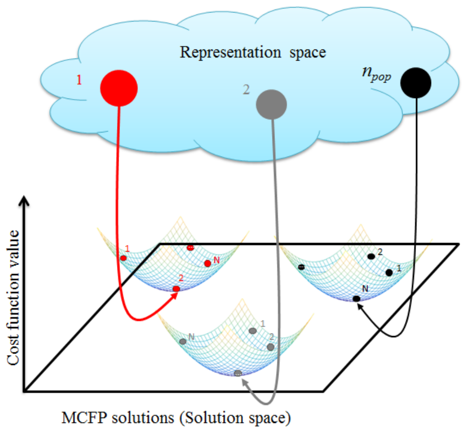


Fig. 4: The local search procedure for GALS (adapted from [36]).

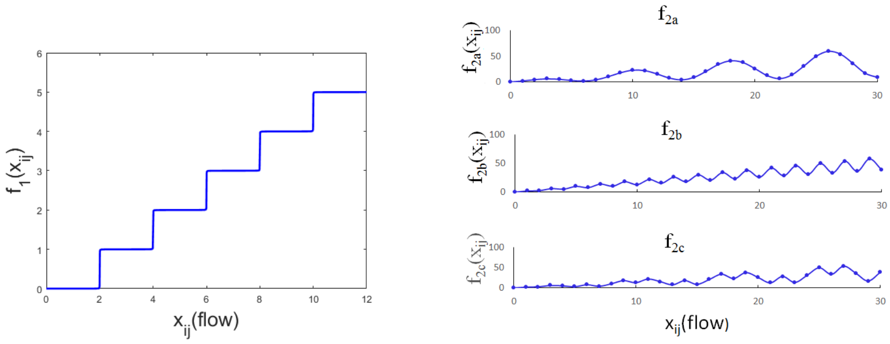
4 Test problems

We focus on evaluating GALS on the single-source single-sink MCFP using nonlinear and non-convex cost functions. Several different types of nonlinear non-convex functions were suggested by Michalewicz [22] on the transportation problems. We select the following nonlinear non-convex cost functions (as shown in Fig. 5) for our study as they are considered to be more practical and challenging than others [22–24]:

$$\begin{aligned} f_1(x_{ij}) =& arctan(PA(x_{ij} - S))/\pi + 0.5+ \\ & arctan(PA(x_{ij} - 2S))/\pi + 0.5+ \\ & arctan(PA(x_{ij} - 3S))/\pi + 0.5+ \\ & arctan(PA(x_{ij} - 4S))/\pi + 0.5+ \\ & arctan(PA(x_{ij} - 5S))/\pi + 0.5, \end{aligned} \tag{5}$$

$$f_2(x_{ij}) = \quad 100 \times \big(x_{ij}(sin\Big(\frac{5\pi x_{ij}^w}{4S}\Big) + 1.3)\big), \tag{6}$$

where the values of PA is set to 1000 and S is set to 2 for f_1, and 5 for f_2, respectively [23]. To examine the robustness of the proposed algorithm, the parameter w in the cost function f_2 is set to 1, 2 and 3, to generate f_{2a}, f_{2b} and f_{2c} functions, respectively. For our evaluation purpose, random networks are created with different sizes from 5 to 100 nodes and with a random number of arcs (decision variables) from 7 to approximately 2500. These network instances are categorised in small, medium-sized (5 to 40 nodes), and large-sized problems (60 to 100 nodes). All these network instances are used for evaluating our proposed algorithm. Our results are compared with those of the commercial mathematical programming solver namely LindoGlobal [35] and the standard GA.



(a) f_1 (b) f_2

Fig. 5: The shape of the cost functions are presented in Eqs. (5) and (6).

4.1 Mathematical solvers

Although exact and heuristic methods exist for solving an MINLP where the objective and constraints are convex, in practice most of the functions are non-convex, which makes the problem extremely difficult to solve [31]. The relaxation of a non-convex MINLP (to make it convex) is itself a global optimisation problem, and it is likely to be NP-hard [32, 33]. Some representative algorithms for solving non-convex MINLPs include spatial branch-and-bound, branch-and-reduce and α branch-and-bound [31]. Based on the aforementioned algorithms, some commercial and open source solvers have been developed for solving non-convex MINLPs, such as CPLEX, Baron, Couenne and LindoGlobal [31].

Nevertheless, these mathematical solver packages have many limitations. For instance, CPLEX is probably one of the most powerful solvers, but it can only handle mixed-integer quadratic programs under certain conditions for constraints and objective functions. Clearly, CPLEX is unable to handle other types of nonlinear functions. The solvers that are able to solve the general non-convex MINLPs include BARON, LindoGlobal, Couenne [31]. BARON [34] is unable to handle trigonometric functions $sin(x)$, $cos(x)$, and Couenne is unable to handle the *arctangent* functions. Among these solvers, LindoGlobal is the only solver that can handle different types of nonlinear functions [35]. Hence in this paper we compare our proposed method GALS with LindoGlobal as well as the standard GA. Unlike GAs, the capability and performance of the mathematical solvers are highly dependent on the shapes of the cost functions adopted in the non-convex MINLPs.

4.2 Numerical results

Our proposed method GALS and the standard GA are implemented in MATLAB on a PC with Intel(R) Core(TM) i7-6500U 2.50 GHz processor with 8 GB RAM and run 30 times for each problem instance using cost function f_1, f_{2a}, f_{2b} and f_{2c}. The computational results for GALS, LindoGlobal and the standard GA, are presented in Tables 1 to 4.

Parameter settings of GALS are as follows: maximum number of iteration (It_{max}=100), population size (n_{pop}=50), number of local search for each individual (N=20), crossover rate (Pc=0.95), mutation rate (Pm=0.3). The parameter settings for the standard GA method are as follows: It_{max}=500, n_{pop}=200, Pc=0.95, Pm=0.3. For both methods, the NFEs and β are set to 150,000 and 20 respectively.

In Tables 1 to 4, for the non-deterministic methods (GALS and standard GA), b, *std* and t denote the best, standard deviation of the results and the average of running time in seconds respectively, and the *mean* represents the average of objective function values over 30 runs and finally h denotes the result of pairwise t-test. For LindoGlobal, the objective function value is recorded in "Obj1" column after t1 seconds and the LindoGlobal keeps running for t2 seconds and the final objective function value is recorded in "Obj2" column. All computational times are reported in seconds.

As shown in Tables 1 to 4, the highest average time for GALS is 286 seconds. To make the results more comparable, we record the objective value found by LindoGlobal at 300s, as reported in "Obj1" column (note that "-" in "t1" and "Obj1" indicates that the results are identical to those in "t2" and "Obj2", in which case LindoGlobal found the global optimum before reaching 300s). We also allow LindoGlobal to keep running until either a global optima is found or the termination time (3,600s) is reached, and the results are reported in the "Obj2" column. As can be seen in Tables 1 to 4, LindoGlobal can only find global optima for small instances (5 and 10 nodes).

In order to compare the performance of GALS and the standard GA, the t-test with the significance level of 0.05 is performed. If GALS is equal, superior or inferior to the standard GA, then h is set to 0, 1 and -1 respectively. We also compare the performance of GALS with LindoGlobal by performing a one-sample t-test with the significance level set to 0.05 and either of the methods has better or equal performance than that of the other is highlighted in bold.

Fig. 6 summarises the results of GALS, the standard GA and LindoGlobal for solving all instances of different sizes using nonlinear non-convex cost functions f_1, f_{2a}, f_{2b} and f_{2c}. As shown in Table 1 and for all 45 problem instances, GALS is superior 39 (87%) times, equal 6 times (13%), and inferior 0 times, when compared with the standard GA. When comparing the mean values of GALS with LindoGlobal on all 45 problem instances (cost function f_1, Table 1), GALS is superior 30 times (67%), equal 11 times (24%) and inferior 4 times (9%). It is noticeable that LindoGlobal cannot find any feasible solutions on all the large-sized problems in 3,600 seconds using cost function f_1.

Table 1: Results of GALS, GA, and LindoGlobal using cost function f_1.

No.	Nodes	Arcs	GALS				GA				h	LINDOGlobal			
			t	b	*mean*	*std*	t	b	*mean*	*std*		t1	Obj1	t2	Obj2
Small and medium-sized instances															
1		7	16	5.0021	**5.0021**	9.11E-16	32	5.0021	5.0021	6.13E-08	0	-	-	2	**5.0021**
2		7	17	5.0021	**5.0021**	9.11E-16	34	5.0021	5.0021	4.15E-13	0	-	-	2	**5.0021**
3	5	8	16	5.0024	**5.0024**	0.00E+00	33	5.0024	5.0024	3.07E-08	0	-	-	1	**5.0024**
4		10	18	5.0032	**5.0032**	1.82E-15	32	5.0032	5.0032	2.73E-12	0	-	-	1	**5.0032**
5		8	18	5.0024	**5.0024**	0.00E+00	33	5.0024	5.0024	1.36E-07	0	-	-	1	**5.0024**
6		32	26	10.0107	**10.0107**	0.00E+00	44	10.2021	10.2023	1.75E-06	1	-	-	5	**10.0107**
7		25	26	10.0081	**10.0081**	3.33E-15	42	10.1461	10.1707	1.93E-02	1	-	-	7	**10.0081**
8	10	34	29	10.0114	**10.0114**	4.66E-15	42	10.2024	10.2036	8.32E-04	1	300	10.0114	949	**10.0114**
9		20	42	10.0063	**10.0063**	2.73E-15	40	10.2103	10.2495	6.97E-03	1	-	-	2	**10.0063**
10		32	40	10.0107	**10.0107**	0.00E+00	46	10.2037	10.2227	3.51E-03	1	300	10.0107	3600	**10.0107**
11		123	64	6.5500	9.2463	1.45E+00	72	9.0463	9.9089	4.18E-01	0	300	6.0494	3600	**6.0490**
12		142	61	4.0571	4.8542	2.76E-01	63	7.5546	8.0226	9.32E-01	1	300	4.0572	3600	**4.0569**
13	20	88	48	10.0310	10.0310	0.00E+00	59	12.0475	12.0475	3.47E-06	1	300	9.5374	3600	**9.0370**
14		133	68	13.5528	15.3900	1.23E+00	70	15.5511	16.6254	8.64E-01	1	300	15.0554	3600	**15.0553**
15		117	63	14.5461	**16.1869**	9.74E-01	64	16.5434	17.0505	7.27E-01	1	300	19.5448	3600	18.5456
16		363	66	10.1311	10.1311	0.00E+00	81	11.2196	12.4407	1.69E-01	1	300	NF	3600	**7.6402**
17		295	64	10.1063	**10.1063**	3.50E-15	85	12.2687	13.2361	2.13E-01	1	300	NF	3600	14.6145
18	40	320	66	10.1154	**10.1154**	2.25E-14	93	11.1039	13.0880	1.99E+00	1	300	NF	3600	12.6241
19		343	69	10.1238	**10.1238**	4.45E-15	95	13.1794	14.0112	6.23E-01	1	300	NF	3600	13.1331
20		294	65	10.1060	**10.1060**	0.00E+00	80	12.1860	13.0349	3.83E-01	1	300	NF	3600	NF
Large-sized instances															
21		844	121	3.8220	**7.5678**	2.67E+00	149	9.3142	13.4360	2.83E+00	1	300	NF	3600	NF
22		905	127	6.8411	**10.0384**	2.10E+00	167	13.3323	15.1815	1.59E+00	1	300	NF	3600	NF
23	60	798	111	6.3024	**9.7742**	2.34E+00	124	11.2962	14.1178	1.65E+00	1	300	NF	3600	NF
24		912	124	4.8475	**9.0931**	2.35E+00	203	12.8363	15.5336	1.54E+00	1	300	NF	3600	NF
25		870	127	3.3310	**6.9530**	2.27E+00	188	10.3215	12.5202	1.25E+00	1	300	NF	3600	NF
26		1176	138	4.4403	**7.3391**	2.25E+00	163	10.4329	13.7305	2.05E+00	1	300	NF	3600	NF
27		1163	146	3.4391	**6.9352**	1.86E+00	169	10.4280	12.5263	2.24E+00	1	300	NF	3600	NF
28	70	1231	152	1.4621	**4.8605**	1.36E+00	189	10.4547	12.7022	1.14E+00	1	300	NF	3600	NF
29		991	148	4.3745	**8.3473**	2.23E+00	174	9.8679	13.5143	1.72E+00	1	300	NF	3600	NF
30		1252	133	3.4695	**6.6675**	2.29E+00	187	10.4619	14.4336	2.79E+00	1	300	NF	3600	NF
31		1718	151	3.1368	**5.6617**	1.83E+00	193	8.1334	11.4305	2.25E+00	1	300	NF	3600	NF
32		1812	142	3.1710	**5.1711**	1.53E+00	144	9.1641	11.7380	2.52E+00	1	300	NF	3600	NF
33	80	1513	133	3.0649	**6.3123**	2.67E+00	194	10.0568	13.7298	1.37E+00	1	300	NF	3600	NF
34		1880	157	3.1957	**4.7464**	1.13E+00	193	9.6892	12.4894	2.56E+00	1	300	NF	3600	NF
35		1619	142	1.6037	**4.0776**	1.24E+00	201	10.0940	12.5183	2.09E+00	1	300	NF	3600	NF
36		1893	165	2.7037	**6.3511**	2.66E+00	209	10.1936	13.3174	2.47E+00	1	300	NF	3600	NF
37		2013	202	3.2470	**5.8454**	2.34E+00	232	11.2367	13.5627	1.92E+00	1	300	NF	3600	NF
38	90	2185	164	2.8087	**5.8079**	2.40E+00	219	9.3009	11.6497	1.25E+00	1	300	NF	3600	NF
39		1944	159	5.2195	**10.0152**	3.22E+00	210	12.2108	14.0598	9.04E-01	1	300	NF	3600	NF
40		2013	171	2.7462	**4.9947**	2.06E+00	277	8.7389	12.9856	2.33E+00	1	300	NF	3600	NF
41		2501	239	1.4249	**5.1976**	2.29E+00	286	9.4174	13.2389	2.95E+00	1	300	NF	3600	NF
42		2512	283	2.4300	**6.2270**	3.54E+00	292	9.4223	14.0197	1.96E+00	1	300	NF	3600	NF
43	100	2437	228	2.4001	**4.5490**	1.20E+00	255	7.8930	10.6174	1.63E+00	1	300	NF	3600	NF
44		2370	279	2.3758	**4.7743**	2.03E+00	310	9.8683	11.0182	5.40E-01	1	300	NF	3600	NF
45		2503	265	2.9247	**8.0457**	3.32E+00	267	10.9191	13.3393	1.42E+00	1	300	NF	3600	NF

To examine if GALS is robust to the different shapes of a cost function, we use cost function f_2 (Eq. 6), choosing three different values for the parameter w in Eq. 6. As shown in Fig. 5b, by increasing the parameter w from 1, to 2 and 3, the number of peaks and valleys (local optima) are increased gradually in functions f_{2b}, f_{2c}. Dealing with these cost functions will be a challenging task. A robust optimisation algorithm should be able to handle this sort of highly non-convex shaped cost functions, without degrading their performances.

As can be seen in Table 2, although LindoGlobal outperformed GALS on some small and medium size problem instances, GALS achieved significantly better performances than those of LindoGlobal on large-sized problems. Additionally, in Tables 3 and 4, GALS significantly outperforms LindoGlobal on all large-sized instances and LindoGlobal has increasing difficulty in finding feasible solutions on instances with 70, 80, 90 and 100 nodes using cost function f_{2b} as well as on instances with 80, 90 and 100 nodes using cost function f_{2c} (except instances No.35,39,40). It is evident that the mathematical solver is sensitive to the non-convex shapes introduced in the cost functions f_{2a}, f_{2b}, and f_{2c}. In contrast, GALS performance is much more robust with respect to

Table 2: Results of GALS, GA, and LindoGlobal using cost function f_{2a}.

No.	Nodes	Arcs	GALS				GA				h	LINDOGlobal			
			t	b	*mean*	*std*	t	b	*mean*	*std*		t1	Obj1	t2	Obj2
Small and medium-sized instances															
1		7	18	9.6000	**9.6000**	2.92E-12	17	26.0000	9.6000	7.29E-15	0	-	-	1	**9.6000**
2		7	16	11.4000	**11.4000**	5.16E-11	16	26.0000	11.4000	7.29E-15	0	-	-	1	**11.4000**
3	5	8	18	9.6000	**9.6000**	6.21E-10	17	26.0000	9.6000	7.29E-15	0	-	-	1	**9.6000**
4		10	19	11.4000	**11.4000**	5.24E-12	19	26.0000	11.4000	7.29E-15	0	-	-	2	**11.4000**
5		8	17	9.6000	**9.6000**	4.86E-14	17	26.0000	9.6000	7.29E-15	0	-	-	1	**9.6000**
6		32	23	32.4853	32.6610	4.29E-01	27	33.6569	33.6569	0.00E+00	1	-	-	5	**21.0000**
7		25	23	29.4355	29.4355	3.65E-15	23	30.6000	30.8828	6.91E-01	1	-	-	42	**29.4284**
8	10	34	25	31.8142	32.1827	7.56E-01	27	33.6569	33.6569	0.00E+00	1	-	-	50	**21.0000**
9		20	30	35.6640	36.1619	6.09E-01	29	45.5431	45.9446	8.24E-01	1	-	-	10	**32.0213**
10		32	27	33.2000	33.5582	7.32E-01	26	34.7147	34.9214	4.29E-01	1	-	-	295	**32.0213**
11		123	65	32.4853	**32.4853**	7.29E-15	66	32.4853	34.6143	1.57E+00	1	300	32.4853	3600	**32.4853**
12		142	75	32.4853	32.4853	4.61E-15	78	32.4853	32.4853	3.65E-15	0	300	30.8569	3600	**30.8569**
13	20	88	70	35.6640	37.6254	2.58E+00	71	38.2569	40.7929	3.36E+00	1	300	28.4938	3600	**28.2000**
14		133	77	75.7218	76.6429	1.06E+00	79	75.8934	77.6869	8.29E-01	1	300	65.9208	3600	**65.3279**
15		117	70	75.0132	76.2276	1.14E+00	74	75.0132	78.0872	2.14E+00	1	300	71.9635	3600	**67.1635**
16		363	83	18.0000	**18.0000**	0.00E+00	94	18.0000	18.0000	0.00E+00	0	300	89.6061	3600	**18.0000**
17		295	81	18.0000	**18.0000**	0.00E+00	82	18.0000	18.0000	0.00E+00	0	300	95.6430	3600	**18.0000**
18	40	320	85	18.0000	**18.0000**	0.00E+00	80	18.0000	18.0000	0.00E+00	0	300	104.5940	3600	**18.0000**
19		343	82	18.0000	**18.0000**	0.00E+00	77	18.0000	18.0000	0.00E+00	0	300	104.7280	3600	**18.0000**
20		294	84	18.0000	**18.0000**	0.00E+00	80	18.0000	18.0000	0.00E+00	0	300	113.4920	3600	**18.0000**
Large-sized instances															
21		844	114	72.4284	**74.1743**	1.10E+00	109	73.8853	76.0575	2.11E+00	1	300	NF	3600	82.7929
22		905	119	72.1563	**72.1563**	1.46E-14	118	73.7990	76.9609	2.21E+00	1	300	NF	3600	72.6274
23	60	798	102	74.9706	75.8439	1.17E+00	104	74.9706	79.2687	2.83E+00	1	300	NF	3600	**71.0711**
24		912	120	74.9706	75.1962	4.07E-01	122	74.9706	78.7940	2.56E+00	1	300	NF	3600	**72.6274**
25		870	115	73.7990	74.2090	5.73E-01	125	73.7990	76.9813	2.53E+00	1	300	NF	3600	**72.6274**
26		1176	140	72.6274	73.4475	9.39E-01	135	73.7990	76.8498	2.00E+00	1	300	NF	3600	**70.2426**
27		1163	134	77.3137	78.3909	1.11E+00	135	77.3137	80.8208	1.80E+00	1	300	NF	3600	**72.6274**
28	70	1231	140	72.6274	**72.6274**	2.92E-14	143	74.9706	77.6185	1.29E+00	1	300	NF	3600	73.7990
29		991	125	75.8934	**77.1638**	1.25E+00	129	77.3137	81.2741	2.01E+00	1	300	NF	3600	**77.3208**
30		1252	135	76.1421	**76.7555**	8.06E-01	133	76.1421	81.2376	2.52E+00	1	300	NF	3600	104.5198
31		1718	154	74.9706	**76.0555**	1.52E+00	148	74.9706	78.5004	1.13E+00	1	300	NF	3600	99.2061
32		1812	158	73.7990	**74.7362**	9.77E-01	159	74.9706	78.7259	1.58E+00	1	300	NF	3600	NF
33	80	1513	137	79.6640	**80.3737**	7.60E-01	133	80.3431	84.3993	2.12E+00	1	300	NF	3600	110.9340
34		1880	165	73.7990	**74.3262**	5.98E-01	161	73.7990	78.8798	2.75E+00	1	300	NF	3600	128.2477
35		1619	155	74.9706	**75.3220**	8.58E-01	157	74.9706	79.7236	2.29E+00	1	300	NF	3600	101.6487
36		1893	161	72.6274	**74.0583**	1.07E+00	165	72.6274	77.0526	2.64E+00	1	300	NF	3600	98.4985
37		2013	188	76.1421	**77.1539**	1.21E+00	180	77.3137	80.3912	1.70E+00	1	300	NF	3600	78.5564
38	90	2185	193	72.6274	73.7236	9.25E-01	187	73.7990	78.6075	2.56E+00	1	300	NF	3600	**72.7990**
39		1944	181	72.7360	**73.7368**	1.06E+00	184	74.4863	78.8511	2.34E+00	1	300	NF	3600	104.2487
40		2013	193	73.7990	74.3186	5.52E-01	195	75.5289	76.8941	1.00E+00	1	300	NF	3600	**68.4061**
41		2501	248	78.1492	**78.6954**	4.19E-01	241	80.3431	82.5764	1.85E+00	1	300	NF	3600	144.4203
42		2512	257	73.7990	**74.2926**	6.52E-01	267	73.7990	78.6355	2.30E+00	1	300	NF	3600	120.4264
43	100	2437	260	77.9421	**78.0140**	2.76E-01	249	79.3208	84.2354	2.20E+00	1	300	NF	3600	124.2335
44		2370	250	77.3137	**77.7539**	8.24E-01	247	77.5431	81.0622	2.43E+00	1	300	NF	3600	83.0345
45		2503	253	75.2203	**76.7672**	1.46E+00	258	78.0000	80.2107	1.93E+00	1	300	NF	3600	126.9980

these cost functions. Note that GALS has superior or equal performance than that of the standard GA on all instances using cost functions f_{2a}, f_{2b} and f_{2c}.

In order to show the convergence speed of GALS compared to the standard GA, the convergence graphs for the large-sized problems are presented in Fig. 7. Since LindoGlobal was unable to find any feasible solution in the first 300 seconds, the result by LindoGlobal is not included in Fig. 7. As can be seen in Fig. 7, GALS is able to converge faster and find better quality solutions than those of the standard GA (Fig. 7).

5 Conclusion

This paper proposes a hybrid genetic algorithm with local search (GALS) for solving a single-source single-sink MCFP. Since many real-world MCFPs cannot be adequately formulated using linear and convex cost functions, in this paper a general nonlinear non-convex single-source single-sink MCFP is considered. The proposed GALS method is compared with the standard GA, and a mathematical solver LindoGlobal. The proposed algorithm has been evaluated on a set of 45 small, medium and large-sized MCFP instances. Our experi-

Table 3: Results of GALS, GA, and LindoGlobal using cost function f_{2b}.

Small and medium-sized instances															
No.	Nodes	Arcs	GALS				GA				h	LINDOGlobal			
			t	b	*mean*	*std*	t	b	*mean*	*std*		t1	Obj1	t2	Obj2
1		7	16	26.0000	**26.0000**	1.29E-15	18	26.0000	26.0000	6.19E-15	0	-	-	1	**26.0000**
2		7	16	26.0000	**26.0000**	3.21E-15	17	26.0000	26.0000	3.39E-15	0	-	-	2	**26.0000**
3	5	8	16	26.0000	**26.0000**	8.30E-15	16	26.0000	26.0000	4.70E-15	0	-	-	4	**26.0000**
4		10	18	26.0000	**26.0000**	6.29E-15	17	26.0000	26.0000	5.14E-15	0	-	-	6	**26.0000**
5		8	17	26.0000	**26.0000**	6.30E-15	18	26.0000	26.0000	8.22E-15	0	-	-	3	**26.0000**
6		32	21	52.0000	**52.0000**	1.36E-14	22	52.0000	52.0000	2.22E-14	0	-	-	60	**52.0000**
7		25	20	52.0000	**52.0000**	1.40E-14	20	52.0000	52.0000	1.36E-14	0	-	-	25	**52.0000**
8	10	34	24	52.0000	**52.0000**	2.33E-14	20	52.0000	52.0000	1.28E-14	0	-	-	40	**52.0000**
9		20	22	52.0000	**52.0000**	3.50E-14	25	52.0000	52.0000	4.40E-14	0	-	-	23	**52.0000**
10		32	23	52.0000	**52.0000**	2.36E-14	24	52.0000	52.0000	3.26E-14	0	-	-	185	**52.0000**
11		123	58	52.0000	**52.0000**	2.42E-14	59	52.0000	52.0000	3.43E-14	0	300	52.0000	3600	**52.0000**
12		142	61	52.0000	**52.0000**	2.46E-15	60	52.0000	52.0000	3.42E-15	0	300	52.0000	3600	**52.0000**
13	20	88	61	52.0000	**52.0000**	1.36E-14	57	52.0000	52.0000	1.25E-14	0	300	52.0000	3600	**52.0000**
14		133	72	109.5208	**110.5929**	2.40E+00	73	110.1137	114.4523	3.52E+00	1	300	121.4710	3600	121.4711
15		117	67	101.5848	**102.2805**	7.15E-01	65	101.5848	105.4304	2.53E+00	1	300	115.6350	3600	115.6345
16		363	86	78.0000	**78.0000**	1.46E-14	81	78.0000	78.0000	2.47E-12	0	300	NF	3600	**78.0000**
17		295	78	78.0000	**78.0000**	3.08E-12	76	78.0000	78.0000	0.00E+00	0	300	78.0000	3600	**78.0000**
18	40	320	79	78.0000	**78.0000**	3.02E-12	81	78.0000	78.0000	2.67E-12	0	300	NF	3600	**78.0000**
19		343	85	78.0000	**78.0000**	1.46E-14	87	78.0000	78.0000	1.46E-14	0	300	NF	3600	**78.0000**
20		294	89	78.0000	**78.0000**	2.47E-12	88	78.0000	78.0000	0.00E+00	0	300	NF	3600	**78.0000**
Large-sized instances															
21		844	98	95.3350	**95.7707**	6.37E-01	91	96.1563	98.2845	2.83E+00	1	300	NF	3600	114.0843
22		905	105	89.3137	**90.8783**	1.42E+00	115	89.3137	94.6350	3.06E+00	1	300	NF	3600	94.7421
23	60	798	94	86.4853	**88.2803**	2.50E+00	95	88.4924	93.0908	4.98E+00	1	300	NF	3600	110.4345
24		912	106	92.1421	**92.8057**	9.44E-01	108	94.1492	97.5341	2.74E+00	1	300	NF	3600	117.5056
25		870	109	89.3137	**89.8291**	6.09E-01	102	89.3137	93.9485	4.40E+00	1	300	NF	3600	165.5401
26		1176	149	86.2569	**86.7861**	9.10E-01	147	86.4853	90.6934	2.40E+00	1	300	NF	3600	NF
27		1163	138	78.0000	**79.3731**	2.45E+00	140	80.8284	85.1097	4.27E+00	1	300	NF	3600	NF
28	70	1231	138	82.8355	**84.0490**	2.50E+00	141	90.4995	95.2600	5.21E+00	1	300	NF	3600	NF
29		991	134	83.4284	**84.2816**	1.78E+00	131	86.4853	91.5123	5.61E+00	1	300	NF	3600	NF
30		1252	151	82.8355	**84.6055**	2.59E+00	155	82.8355	91.7081	5.41E+00	1	300	NF	3600	NF
31		1718	165	86.2569	**86.7747**	1.13E+00	170	87.0782	94.2814	6.08E+00	1	300	NF	3600	NF
32		1812	160	86.4853	**86.5856**	4.49E-01	174	88.4924	92.1623	2.47E+00	1	300	NF	3600	NF
33	80	1513	165	87.0782	**87.8103**	1.85E+00	154	88.4924	99.3220	7.58E+00	1	300	NF	3600	NF
34		1880	169	86.4853	**87.0578**	1.03E+00	173	89.3137	95.9374	6.38E+00	1	300	NF	3600	NF
35		1619	155	82.8355	**85.2283**	9.57E-01	163	83.6569	89.5324	5.45E+00	1	300	NF	3600	NF
36		1893	166	80.8284	**83.5290**	5.17E+00	169	82.8355	91.3799	5.68E+00	1	300	NF	3600	NF
37		2013	193	86.4853	**87.8949**	2.43E+00	190	91.9137	100.5540	7.52E+00	1	300	NF	3600	NF
38	90	2185	202	88.4924	**89.2656**	1.34E+00	195	88.4924	98.7726	7.61E+00	1	300	NF	3600	NF
39		1944	188	83.6569	**84.5350**	1.38E+00	189	83.6569	90.5884	5.22E+00	1	300	NF	3600	NF
40		2013	195	82.8355	**84.1494**	2.38E+00	200	83.6569	90.0845	2.90E+00	1	300	NF	3600	NF
41		2501	251	85.6640	**87.2607**	4.65E+00	229	86.4853	95.8368	7.43E+00	1	300	NF	3600	NF
42		2512	270	91.3208	**92.5593**	1.57E+00	262	92.1421	97.7596	4.24E+00	1	300	NF	3600	NF
43	100	2437	278	82.8355	**84.0673**	2.02E+00	267	83.6569	97.4563	1.00E+01	1	300	NF	3600	NF
44		2370	285	86.2569	**86.6218**	1.09E+00	288	90.4995	97.3263	5.72E+00	1	300	NF	3600	NF
45		2503	286	95.3350	**96.1425**	1.48E+00	265	95.3350	104.2375	7.78E+00	1	300	NF	3600	NF

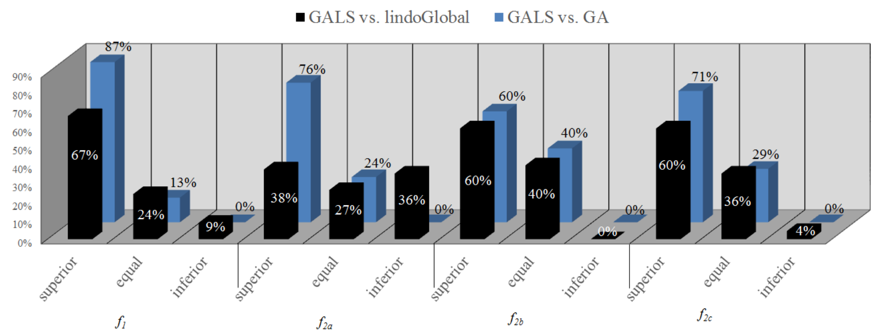
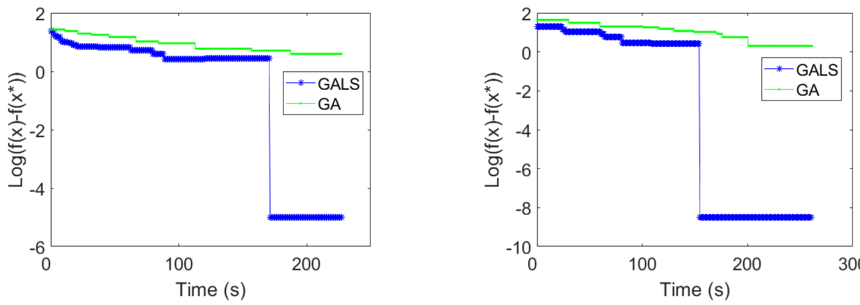


Fig. 6: The number of "wins-draws-loses" of GALS as compared with that of standard GA and LindoGlobal using cost functions f_1, f_{2a}, f_{2b} and f_{2c}.

mental results show that GALS method significantly outperforms the standard GA and LindoGlobal in terms of solution quality and convergence speed. It is clearly evident that GALS method can handle the large-sized MCFP instances more effectively and efficiently. In contrast, LindoGlobal could not find any feasible solutions for large-sized problems using cost function f_1, f_{2b} and f_{2c}. In

Table 4: Results of GALS, GA, and LindoGlobal using cost function f_{2c}.

No.	Nodes	Arcs	GALS t	GALS b	GALS *mean*	GALS *std*	GA t	GA b	GA *mean*	GA *std*	*h*	LINDOGlobal t1	LINDOGlobal Obj1	LINDOGlobal t2	LINDOGlobal Obj2
Small and medium-sized instances															
1		7	19	17.7868	**17.7868**	3.54E-14	20	26.0000	17.7868	0.00E+00	0	-	-	1	**17.7868**
2		7	17	20.7513	**20.7513**	3.91E-14	18	26.0000	20.7513	0.00E+00	0	-	-	1	**20.7513**
3	5	8	17	17.7868	**17.7868**	2.48E-10	18	26.0000	17.7868	0.00E+00	0	-	-	2	**17.7868**
4		10	19	20.7513	**20.7513**	1.56E-09	15	26.0000	20.7513	0.00E+00	0	-	-	29	**20.7513**
5		8	18	17.7868	**17.7868**	3.85E-11	16	26.0000	17.7868	0.00E+00	0	-	-	2	**17.7868**
6		32	27	28.9157	**32.7254**	2.91E+00	27	35.0294	35.6087	1.19E+00	1	300	32.2010	3600	**32.2010**
7		25	22	31.9726	**31.9726**	1.09E-14	27	31.9726	31.9726	1.09E-14	0	300	31.9729	3600	31.9729
8	10	34	29	32.2010	**32.2010**	0.00E+00	29	35.0294	35.1480	2.43E-01	1	300	32.2010	3600	**32.2010**
9		20	35	32.2010	**32.2010**	0.00E+00	38	43.6508	43.6508	0.00E+00	1	300	32.2010	3600	**32.2010**
10		32	31	32.2010	**32.2010**	1.46E-14	34	32.2010	32.2010	1.46E-14	0	300	32.2010	3600	**32.2010**
11		123	68	32.2010	**32.2010**	6.10E-15	65	32.2010	33.8227	1.55E+00	1	300	32.2010	3600	**32.2010**
12		142	73	35.0294	35.2962	3.03E-01	74	37.0365	37.8076	6.88E-01	1	300	32.2010	3600	**32.2010**
13	20	88	70	34.2081	**34.9881**	1.06E+00	71	35.6223	39.4494	3.26E+00	1	300	34.8010	3600	**34.8010**
14		133	74	65.5442	**65.5442**	2.92E-14	75	66.9584	69.0044	1.32E+00	1	300	108.5580	3600	81.8294
15		117	67	73.8934	73.8934	1.46E-14	77	73.8934	75.5562	1.40E+00	1	300	93.6000	3600	**72.6152**
16		363	77	38.4020	**38.4020**	1.46E-14	70	38.4020	38.9767	2.14E+00	0	300	NF	3600	**38.4020**
17		295	65	38.4020	**38.4020**	1.46E-14	67	38.4020	38.4020	1.46E-14	0	300	NF	3600	**38.4020**
18	40	320	71	38.4020	**38.4020**	1.46E-14	72	38.4020	38.5024	4.49E-01	0	300	NF	3600	**38.4020**
19		343	78	38.4020	**38.7031**	9.82E-01	74	38.4020	38.6027	6.18E-01	0	300	NF	3600	**38.4020**
20		294	72	38.4020	**38.4020**	1.46E-14	74	38.4020	39.0042	1.47E+00	0	300	NF	3600	**38.4020**
Large-sized instances															
21		844	100	71.4294	**71.8355**	5.75E-01	109	72.6152	75.2089	2.84E+00	1	300	NF	3600	118.5553
22		905	114	61.0294	**61.9052**	1.23E+00	112	63.8579	68.0798	2.13E+00	1	300	NF	3600	225.0630
23	60	798	106	61.0294	**62.1858**	1.34E+00	104	61.0294	64.4554	2.27E+00	1	300	NF	3600	96.2924
24		912	124	63.8579	**64.2411**	9.48E-01	120	65.8650	67.8427	1.82E+00	1	300	NF	3600	87.0782
25		870	104	62.8081	**63.2919**	5.28E-01	107	63.6294	66.2392	1.91E+00	1	300	NF	3600	219.4061
26		1176	135	58.2010	**59.7566**	1.44E+00	110	61.0294	62.0376	1.51E+00	1	300	NF	3600	77.7716
27		1163	120	58.2010	**58.8967**	1.24E+00	116	58.2010	59.8456	2.41E+00	0	300	NF	3600	90.4071
28	70	1231	141	58.2010	**59.4692**	2.08E+00	148	63.0365	66.6746	2.51E+00	1	300	NF	3600	75.1716
29		991	124	58.2010	**59.7224**	1.41E+00	121	58.2010	62.7219	2.53E+00	1	300	NF	3600	90.4071
30		1252	148	58.2010	**60.2652**	1.88E+00	149	58.2010	66.4671	3.29E+00	1	300	NF	3600	91.5929
31		1718	156	58.2010	**60.0759**	2.47E+00	147	63.0365	67.7377	3.34E+00	1	300	NF	3600	NF
32		1812	152	58.2010	**59.9984**	1.57E+00	164	60.2081	64.9640	2.63E+00	1	300	NF	3600	NF
33	80	1513	147	58.2010	**59.4624**	2.03E+00	160	63.0365	66.2047	2.35E+00	1	300	NF	3600	NF
34		1880	166	60.8010	**62.2405**	1.43E+00	160	63.0365	66.4603	2.37E+00	1	300	NF	3600	NF
35		1619	155	58.2010	**58.6139**	1.01E+00	165	58.2010	60.9359	2.71E+00	1	300	NF	3600	186.3787
36		1893	165	58.2010	**59.3802**	1.81E+00	167	58.2010	62.4687	2.54E+00	1	300	NF	3600	NF
37		2013	198	58.2010	**59.3324**	2.32E+00	201	63.8579	67.4959	2.37E+00	1	300	NF	3600	NF
38	90	2185	204	60.8010	**61.2117**	8.10E-01	210	63.0365	66.3439	2.46E+00	1	300	NF	3600	NF
39		1944	197	58.2010	**59.6152**	1.45E+00	191	58.2010	61.2598	2.14E+00	1	300	NF	3600	83.4721
40		2013	210	58.2010	**60.2402**	2.54E+00	214	60.2081	64.3482	2.19E+00	1	300	NF	3600	239.2975
41		2501	252	61.0294	**62.7926**	1.72E+00	248	61.0294	67.2267	3.49E+00	1	300	NF	3600	NF
42		2512	263	65.8650	**66.6382**	8.95E-01	253	65.8650	69.6513	3.03E+00	1	300	NF	3600	NF
43	100	2437	261	58.2010	**59.3095**	1.61E+00	263	58.2010	64.9343	4.03E+00	1	300	NF	3600	NF
44		2370	264	60.8010	**61.8527**	1.68E+00	271	61.0294	65.4772	3.29E+00	1	300	NF	3600	NF
45		2503	274	61.6223	**62.5712**	1.16E+00	278	70.2437	71.9140	1.65E+00	1	300	NF	3600	NF



(a) f_{2a} on instance No.45 (b) f_{2c} on instance No.40

Fig. 7: Convergence plot for the proposed GALS method and standard GA.

addition, GALS can handle very well the selected nonlinear non-convex cost functions, whereas existing mathematical solvers are too sensitive to the shape of the function. This robustness property is an important strength of the GA-based methods in solving MCFPs. Our future work will examine performance of GALS method on practical telecommunication network problems.

6 Compliance with Ethical Standards

Conflict of Interest: The authors declare that they have no conflict of interest.

Ethical approval: This article does not contain any studies with human participants or animals performed by any of the authors.

References

1. Ahuja, R.K., Magnanti, T.L., and Orlin, J.B., Network flows: theory, algorithms, and applications,Prentice hall , pp.4–6, (1993).
2. Vegh, Laszlo A. A strongly polynomial algorithm for a class of minimum-cost flow problems with separable convex objectives." SIAM Journal on Computing 45.5 (2016): 1729-1761.
3. Yan, S., Y. L. Shih, and C. L. Wang. An ant colony system-based hybrid algorithm for square root concave cost transhipment problems. Engineering Optimization 42.11 (2010): 983-1001.
4. Lin, ShinYeu, and ChiHsin Lin. A computationally efficient method for nonlinear multicommodity network flow problems. Networks: An International Journal 29.4 (1997): 225-244.
5. Newell, G. F. Non-convex traffic assignment on a rectangular grid network. Transportation science 30.1 (1996): 32-42.
6. Garey, Michael R., and David S. Johnson. Computers and intractability: A guide to the theory of npcompleteness (series of books in the mathematical sciences), ed. Computers and Intractability 340 (1979).
7. Guisewite, Geoffrey M., and Panos M. Pardalos. Algorithms for the single-source uncapacitated minimum concave-cost network flow problem. Journal of Global Optimization 1.3 (1991): 245-265.
8. Monteiro, Marta SR, Dalila BMM Fontes, and Fernando ACC Fontes. Concave minimum cost network flow problems solved with a colony of ants. Journal of Heuristics 19.1 (2013): 1-33.
9. Kim, Hak-Jin, and John N. Hooker. Solving fixed-charge network flow problems with a hybrid optimization and constraint programming approach. Annals of Operations Research 115.1-4 (2002): 95-124.
10. Ortega, Francisco, and Laurence A. Wolsey. A branchandcut algorithm for the singlecommodity, uncapacitated, fixedcharge network flow problem. Networks: An International Journal 41.3 (2003): 143-158.
11. Fontes, Dalila BMM, Eleni Hadjiconstantinou, and Nicos Christofides. A branch-and-bound algorithm for concave network flow problems. Journal of Global Optimization 34.1 (2006): 127-155.
12. Guisewite, Geoffrey M., and Panos M. Pardalos. Global search algorithms for minimum concave-cost network flow problems. Journal of Global Optimization 1.4 (1991): 309-330.
13. Horst, Reiner, and Nguyen V. Thoai. An integer concave minimization approach for the minimum concave cost capacitated flow problem on networks. Operations-Research-Spektrum 20.1 (1998): 47-53.
14. Burkard, Rainer E., Helidon Dollani, and Phan Thien Thach. Linear approximations in a dynamic programming approach for the uncapacitated single-source minimum concave cost network flow problem in acyclic networks. Journal of Global Optimization 19.2 (2001): 121-139.

15. Fontes, Dalila BMM, Eleni Hadjiconstantinou, and Nicos Christofides. A dynamic programming approach for solving single-source uncapacitated concave minimum cost network flow problems. European Journal of Operational Research 174.2 (2006): 1205-1219.
16. Erickson, Ranel E., Clyde L. Monma, and Arthur F. Veinott Jr. Send-and-split method for minimum-concave-cost network flows. Mathematics of Operations Research 12.4 (1987): 634-664.
17. Kovacs, Peter. Minimum-cost flow algorithms: an experimental evaluation. Optimization Methods and Software 30.1 (2015): 94-127.
18. Monteiro, Marta SR, Dalila BMM Fontes, and Fernando ACC Fontes. An ant colony optimization algorithm to solve the minimum cost network flow problem with concave cost functions. Proceedings of the 13th annual conference on Genetic and evolutionary computation (2011): 139-146.
19. Fontes, Dalila BMM, and Jos Fernando Gonalves. Heuristic solutions for general concave minimum cost network flow problems. Networks: An International Journal 50.1 (2007): 67-76.
20. Xie, Fanrong, and Renan Jia. Nonlinear fixed charge transportation problem by minimum cost flow-based genetic algorithm. Computers and Industrial Engineering 63.4 (2012): 763-778.
21. Reca, Juan, Juan Martnez, and Rafael Lpez-Luque. A new efficient bounding strategy applied to the heuristic optimization of the water distribution networks design. Congress on Numerical Methods in Engineering CMN. 2017.
22. Michalewicz, Zbigniew, George A. Vignaux, and Matthew Hobbs. A nonstandard genetic algorithm for the nonlinear transportation problem. ORSA Journal on computing 3.4 (1991): 307-316.
23. Klansek, Uros, and Mirko Psunder. Solving the nonlinear transportation problem by global optimization. Transport 25.3 (2010): 314-324.
24. Klansek, Uros. Solving the nonlinear discrete transportation problem by MINLP optimization. Transport 29.1 (2014): 1-11.
25. Goldberg, David E., and John H. Holland. Genetic algorithms and machine learning. Machine learning 3.2 (1988): 95-99.
26. Michalewicz, Zbigniew and Hartley, Stephen. Genetic algorithms+ data structures= evolution programs. Mathematical Intelligence 18.3 (1996).
27. Cheng, Runwei, and Mitsuo Gen. An evolution programme for the resource-constrained project scheduling problem. International Journal of Computer Integrated Manufacturing 11.3 (1998): 274-287.
28. Gen, Mitsuo, Runwei Cheng, and Dingwei Wang. Genetic algorithms for solving shortest path problems. Evolutionary Computation, 1997., IEEE International Conference on. IEEE, 1997.
29. Lin, Lin, and Mitsuo Gen. Multiobjective Genetic Algorithm for Bicriteria Network Design Problems. Intelligent and Evolutionary Systems. Springer, Berlin, Heidelberg, 2009. 141-161.
30. Gen, Mitsuo, Runwei Cheng, and Lin Lin. Network models and optimization: Multiobjective genetic algorithm approach. Springer Science and Business Media, 2008.
31. Burer, Samuel, and Adam N. Letchford. Non-convex mixed-integer nonlinear programming: A survey. Surveys in Operations Research and Management Science 17.2 (2012): 97-106.
32. Tawarmalani, Mohit, Nikolaos V. Sahinidis, and Nikolaos Sahinidis. Convexification and global optimization in continuous and mixed-integer nonlinear programming: theory, algorithms, software, and applications. Springer Science & Business Media (65), (2002).
33. Sherali, Hanif D., and Warren P. Adams. A reformulation-linearization technique for solving discrete and continuous nonconvex problems. Vol. 31. Springer Science and Business Media, 2013.
34. Tawarmalani, Mohit, Nikolaos V. Sahinidis, and Nikolaos Sahinidis. Convexification and global optimization in continuous and mixed-integer nonlinear programming: theory, algorithms, software, and applications. Vol. 65. Springer Science and Business Media, 2002.
35. Lin, Youdong, and Linus Schrage. The global solver in the LINDO API. Optimization Methods and Software 24.4-5 (2009): 657-668.
36. Sinha, Ankur, Pekka Malo, and Kalyanmoy Deb. Evolutionary algorithm for bilevel optimization using approximations of the lower level optimal solution mapping. European Journal of Operational Research 257.2 (2017): 395-411.